Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Cost-sensitive learning classification strategy for predicting product failures

Flavia Dalia Frumosu[a,*], Abdul Rauf Khan[a], Henrik Schiøler[b], Murat Kulahci[a,c], Mohamed Zaki[d], Peter Westermann – Rasmussen[e]

a. Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, 2800 Kongens Lyngby, Denmark
b. Department of Electronic Systems, Aalborg University, Fredrik Bajers Vej 7, 9220 Aalborg, Denmark
c. Department of Business Administration, Technology and Social Sciences, Luleå University of Technology, 97187 Luleå, Sweden
d. Department of Engineering, University of Cambridge, 17 Charles Babbage Rd, Cambridge CB3 0FS, United Kingdom
e. Danfoss Drives A/S, Ulsnaes 1, 6300 Gråsten, Denmark

ARTICLE INFO

Article history:
Received X

Keywords:
Cost-sensitive learning
Predictive manufacturing
Failure prediction
Imbalance classification
Genetic algorithm
Voronoi diagram

ABSTRACT

In the current era of Industry 4.0, sensor data used in connection with machine learning algorithms can help manufacturing industries to reduce costs and to predict failures in advance. This paper addresses a binary classification problem found in manufacturing engineering, which focuses on how to ensure product quality delivery and at the same time to reduce production costs. The aim behind this problem is to predict the number of faulty products, which in this case is extremely low. As a result of this characteristic, the problem is reduced to an imbalanced binary classification problem. The authors contribute to imbalanced classification research in three important ways. First, the industrial application coming from the electronic manufacturing industry is presented in detail, along with its data and modelling challenges. Second, a modified cost-sensitive classification strategy based on a combination of Voronoi diagrams and genetic algorithm is applied to tackle this problem and is compared to several base classifiers. The results obtained are promising for this specific application. Third, in order to evaluate the flexibility of the strategy, and to demonstrate its wide range of applicability, 25 real-world data sets are selected from the KEEL repository with different imbalance ratios and number of features. The strategy, in this case implemented without a predefined cost, is compared with the same base classifiers as those used for the industrial problem.

1. Introduction

As a result of technological developments and globalization of the world's economy, the manufacturing industry is currently going through a digital transformation phase at a fast pace. The concepts of the Industry 4.0 revolution (Lasi et al., 2014) and the Industrial Internet of Things (IIOT) (Atzori, Iera, & Morabito, 2010) are leading to a faster digitalization of manufacturing (Xiong, & Yin, 2006) and consequently are attracting significant attention from the industry. The main thread running through these concepts is not only the idea of machine-to-machine (M2M) communication, but also the deployment of smart machines in manufacturing (Lee, Lapira, Bagheri, & Kao, 2013); or, in other words, machines capable of making informed, automated and smarter decisions. From a data perspective, this can be achieved through a strategy of handling the sensor data and generating insightful analysis of this data. On the one hand, these analytical capabilities will provide extra insights into the hidden characteristics of the manufacturing process; and, on the other hand, they will facilitate optimization of the production process through more informed and timely decision-making.

The emerging concept of predictive manufacturing is influenced by all these concepts. In a very broad sense, predictive manufacturing refers to intelligent handling of sensor streams and the extraction of actionable insights from these

* Corresponding author. Tel.: +45 45255351
E-mail addresses: fdal@dtu.dk (F.D. Frumosu), arkh@dtu.dk (A.R. Khan), henrik@es.aau.dk (H. Schiøler), muku@dtu.dk (M. Kulahci), mehyz2@cam.ac.uk (M. Zaki), peter.westermann@danfoss.com (P. Westermann-Rasmussen).

heterogeneous data streams (Lee, Lapira, Yang, & Kao, 2013; Krumeich, Jacobi, Werth, & Loos, 2014), with the ultimate goal of using these insights for optimization of the production line.

High volumes of data are commonly collected in manufacturing, and the quality control (QC) stages are one of the key contributors to these information-heavy and heterogeneous data sets. First, heterogeneity in production data is a result of diversity in operations; for example, the quality control stages are normally designed to examine a diverse set of characteristics of the product (Taguchi, 1986). In other words, it is quite possible that in a single process, one QC stage tests product functionality by comparing standard limits (for a particular product), and in the next QC stage, images are analyzed to examine their physical appearance. Second, the existence of missing information can be due to customized production, or any other reason such as data-collection policy or malfunctions, among others. These are just some of the challenges imposed by the production data, which naturally make it more difficult to analyze. Due to the complexity, traditional statistical methods or other data-driven approaches also fail in this context. This research contributes to resolving some of the analytical challenges related to binary data classification problems. We used a large stream of unstructured production data from a world-leading manufacturer of frequency inverter drives. There are different types of challenge associated with this problem, from both the data characteristics and the classifiers' perspective. The industrial data is high dimensional and is severely imbalanced, as in the last quality stage the number of faulty products is extremely low. The first step in our analysis is to perform feature engineering in order to avoid the curse of dimensionality. The resulting data from the feature engineering process is then used for failure prediction using a modified classification strategy based on the work carried out by Khan, Schiøler, Zaki, and Kulahci (2018). From an industrial perspective, we are interested in obtaining a decision rule for going through or skipping elements of the last QC stage (unit test). For this, we create the problem as a trade-off between cost and quality, where our industrial partner specifies the cost associated with a faulty product sent to the client. To assess how well the modified strategy performs on the industrial data set, 10 other classifiers are selected for a comparison study. Since the industrial problem is defined as a cost problem where the cost is specified in advance, we also want to show that the modified strategy is flexible enough for problems when the cost is not available. In this case, we select 25 real-world KEEL data sets with different imbalance ratios to assess how the modified strategy performs.

Contribution wise, the current article extends the work of Khan, Schiøler, Zaki, and Kulahci (2018) in terms of methodology and data applicability, with the mention that the main goal of the entire work is industrial applicability.

The article is arranged as follows. The next section, Section 4, is devoted to related work, while the industrial problem is presented in detail in Section 2. The proposed classification strategy is discussed in Section 5. The solution of the addressed industrial problem is discussed from both a methodological perspective and also from a results-wise perspective in Section 6. A comparison study using 25 KEEL data sets is performed and discussed in Section 7. We conclude the article with a conclusion and final remarks, which are presented in Section 8.

2. Industrial problem

This section aims to describe the industrial process in detail, as well as the data characteristics and problems associated with it.

2.1. General formulation of the industrial problem

In its very general form, a multi-stage process can be seen as a combination of subsequent production stages/operations connected together to fabricate the final product (as presented in Fig. 1). The horizontal axis in Fig. 1 represents different stages of production, where the depth of the process (vertical axis) represents the task designation inside the production process. Fig. 1 is a generic illustration of a typical multi-stage (or sequential) process in which different production/assembly lines (under one roof or multiple roofs) are used to fabricate the final product. Let us consider a multi-stage manufacturing process, as presented in Fig. 1, with a finite number (F) of production stages, where S_i with $i = 1, \dots, F$ represents the outcome of the i_{th} production stage. By the definition of multi-stage manufacturing processing, it is quite straightforward to conclude that the outcomes, S_i, may depend on the outcomes of the subsequent stages $(S_{i-1}, S_{i-2}, \dots, S_{i-(F-1)})$. This implies that:

$$P\left(\bigcap_{i=1}^{F} S_i\right) \neq \prod_{i=1}^{F} P(S_i) \tag{1}$$

For the data analysis purpose, consider S_i as the data-generation step, which can be used further on to predict the outcome of the last stage, S_F, i.e. the final product, which is in fact the general formulation of the industrial problem.

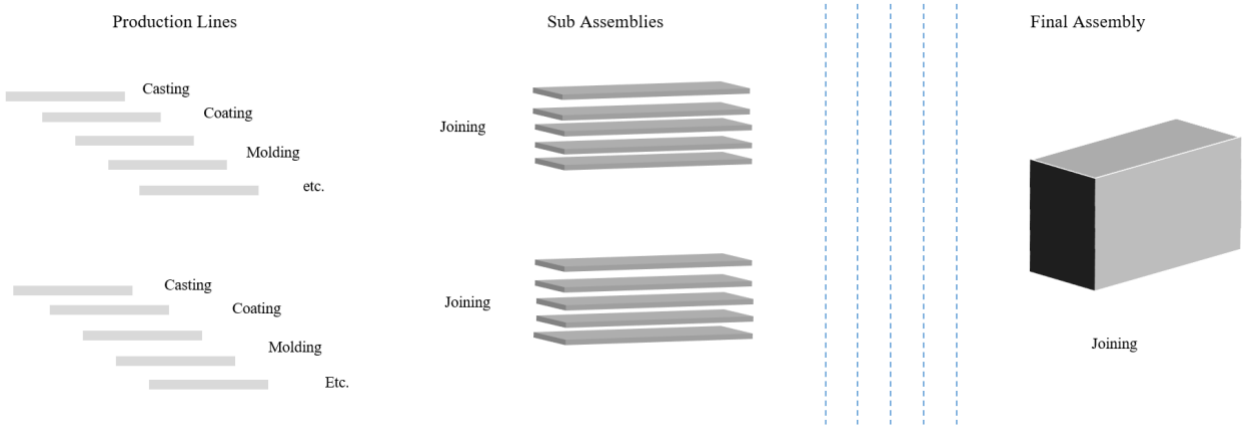


Fig. 1 Multi-stage manufacturing process

2.2. Description of the industrial set-up

The real-life problem deals with the production of frequency inverter drives. The manufacturing process for a frequency inverter drive is a multi-stage manufacturing process, Fig. 1, where the end product is a combination of multiple parts and sub-assemblies produced and assembled in different production stages (in-house and outsourced). A frequency inverter drive comprises electronic components and software, to control and regulate the speed of electric motors. As hinted above, the process under consideration is electronics manufacturing. A primary production task is the production of PCBAs (printed circuit-board assemblies), and the end product typically comprises a combination of two to three PCBAs, depending on customer requirements, such as power, application area and connectivity.

In a broad sense, we can divide this manufacturing process into two phases: 1) the PCBA manufacturing; and 2) the unit assembly, as presented in Fig. 2. In the first phase of the manufacturing, some of the basic operations for PCBA manufacturing (such as screen printing, SMT (surface-mounted technology) and THT (through-hole technology) population and soldering processes) are carried out according to the specific design of the product, whereas the second phase is focused on the assembly-related tasks and installation of the required software.

An important feature of this process is the number of quality control tests after each major modification in the product, as highlighted by the yellow boxes in Fig. 2. Each yellow box represents a quality control stage, where each may contain up to 5,000 quality control tests (or steps) to test different aspects of the product. In the process presented in Fig. 2 there are four quality control stages, named AOI (automated optical inspection), FT (function test), hi-pot test (insulation test) and unit test (the final quality control test). Next, we will briefly discuss these four quality control stages and the general purpose of these stages, as this will facilitate an understanding of the structure, as well as type, of the data used for the analysis.

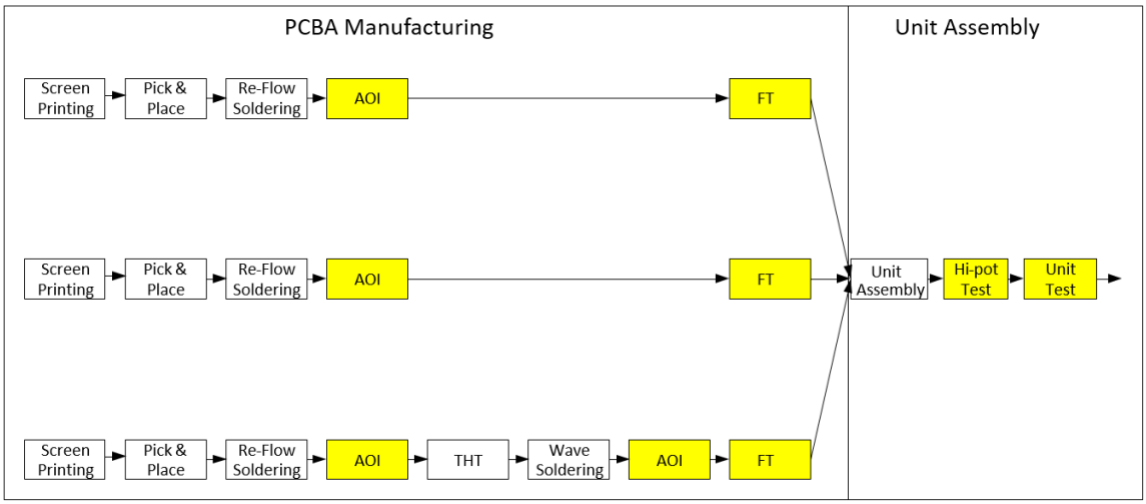


Fig. 2 Pictorial illustration of the manufacturing process

2.3. Industrial data set

The product selected for the purpose of this analysis contains two sub-assemblies (printed circuit-board assemblies – or PCBAs), as presented in Fig. 3 (control card and power card). The first quality control step for each PCBA, regardless of its composition, is the AOI stage. At this stage of the process, we collect image data, as well as the categorical output about the status of the product (as presented by branches 1.a and 2.a in Fig. 3). After passing the AOI stage(s), the product is submitted to the FT stage. FT is a combination of standard, as well as especially designed, test steps to examine the functionality of the PCBAs. As shown in 1.b and 2.b in Fig. 3, it produces a bulk of categorical and continuous outputs. After passing quality control tests at the PCBA manufacturing phase, the completed PCBAs move to the unit assembly phase. The first quality control after unit assembly is the hi-pot test, which tests that the product is electrically safe to use and compliant with the applicable safety standards. Like the function test stage, the hi-pot test contains information in both categorical and continuous formats. The unit test is the last and final quality-assurance test in the manufacturing process; after passing the unit test, the product is ready for delivery.

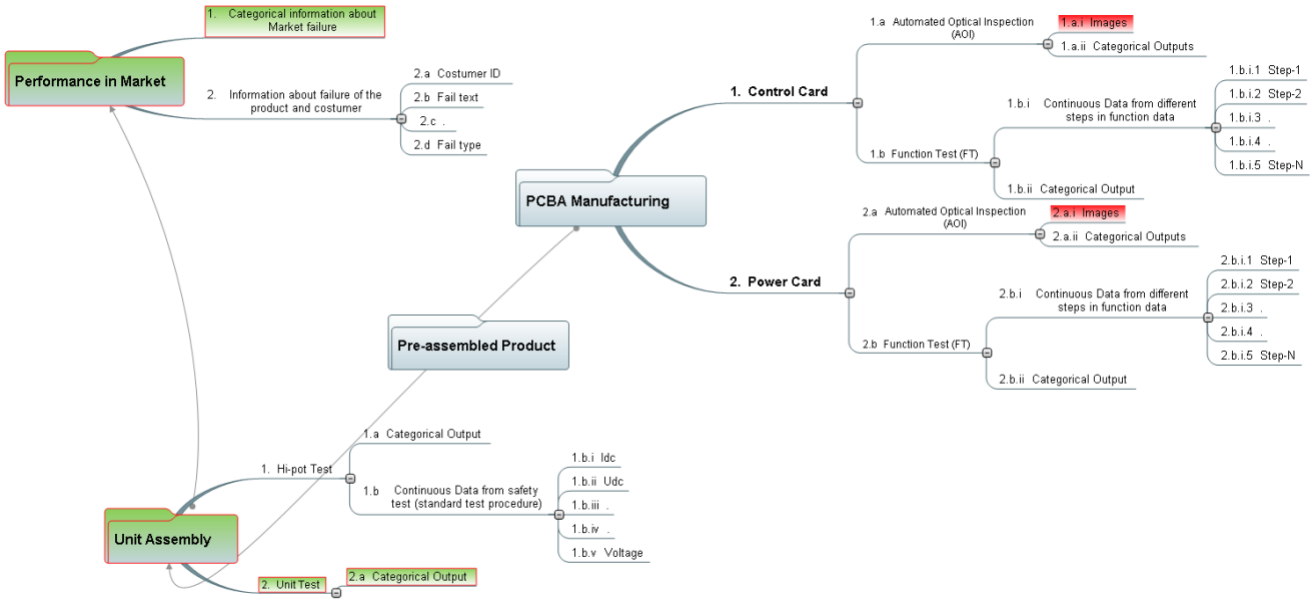


Fig. 3 Production test data structure

For our analysis, we use the information from all three previous stages (hi-pot, function and AOI) for the two sub-assemblies, with the exception of the image data (highlighted red in Fig. 3) to predict last-stage failures (unit test outcome). The data characteristics of the test stages are presented in detail in Table 1.

Table 1
Characteristics of the data set (per one sub-assembly)

Test stage	Type	Dimensions
AOI	Binary	1
Function test	Continuous, Binary	Up to 5,000
High voltage	Continuous, Binary	Up to 10
Unit	Binary	1

The final data set used for the analysis is composed of two data sets, namely, Sub Assembly 1 (control card) data, which has 67,148 samples (products) x 5,021 features, and data from Sub Assembly 2 (power card), which has 67,148 samples (products) x 5,072 features. It is important to mention that the sub-assembly data has a high number of missing values, which is another important characteristic of the data. Without going into a detailed discussion of the root causes, sometimes missing observations inherit some knowledge about the process dynamics. As an example, a missing value in the production data can be due to a special production strategy or quality control strategy. In particular, in the case of quality control testing data, skipping one test may have a significant impact on the quality of the end product.

The final testing status from merging the control and power card is also given as a binary vector of length 67,148, from

which 922 are faulty products and 66,226 are good ones. This means that the classes are distributed as 1% (faulty) / 99% (good), or in other words there is an imbalance ratio of 66,226/922 = 71.83 in the received data set.

3. Problem motivation

This paper addresses the above mentioned industrial problem where the manufacturing process is highly optimized. For analysis purposes, historical quality control data with pass and fail labels is used. As mentioned previously, the data that is received is highly imbalanced, with approximately 99% of the data belonging to the pass class, while the remaining 1% belongs to the fail class. From a machine learning perspective, this makes the problem an imbalanced one with binary classes.

The business case lies in predicting the fail class, which in fact are faulty products and potentially can be sent to the client. Several costs can be associated with this unwanted scenario, e.g. warranty, customer service, travel, reputation costs, and the list goes on; here, we make a tacit assumption that all internal costs are manageable. In the current industrial setting the customer feedback or requirements have not been considered. One way to avoid this scenario is manually taking each product/item and checking it individually, which will ensure that most (if not all) of the faulty products will be caught in advance before they are sent to the client. Evidently, this entire process is extremely costly if there is a high number of products/items to check.

Misclassification of faulty products is costly for the big manufacturing facility and thus, this study aims to find a flexible classification method that can consider both the associated cost and the "quality" that should be sent to customers. In this context, the "quality" is measured as the percentage/number of faulty products that are sent to the customer. As in many cases, there is a trade-off between the associated costs and quality sent to a customer, and usually we expect that a high cost is an indicator of good "quality".

As discussed by Haixiang et al. (2017), there are hundreds of algorithms dealing with imbalanced data classification. Naturally, all of the methods have their strengths and weaknesses, depending on the application and context. In our work, we chose a cost-sensitive learning strategy, since we can incorporate the cost into the analysis, as this is an important aspect for industrial manufacturing firms. The binary classification problem is also highly imbalanced with class overlap and, thus, we decided to resort to the strategy proposed by Khan, Schiøler, Zaki, and Kulahci (2018), which takes into consideration both the imbalance and the overlap (noise) and is flexible enough to include the cost. The strategy relies on a non-parametric discretization of the feature space and subsequent combinatorial optimization using a designed separation statistic, which can be used as a quality measure for assessing the separation between the classes. Furthermore, as a result of the property of feature space discretization, the strategy is also helpful in cases where small disjuncts are found.

4. Related Work

The following section provides a general overview of the previous work carried out in the manufacturing industry, with a focus on quality control, as well as previous work from the area of imbalanced data classification.

4.1. Previous work in the area of manufacturing industry

Failure prediction at system level is a relatively well-developed and well-researched area in manufacturing, with different titles and tags such as reliability analysis and reliability prediction. Different data-driven methods such as support vector machines (SVM) (das Chagas Moura, Zio, Lins, & Droguett, 2011), rough set theory (Kusiak, 2001), sensitivity analysis (Iannuzzelli, 1991), and many others, as presented in Denson (1998), O'Connor and Kleyner (2011), and Blischke, and Murthy (2011), form a part of the reliability literature. Historically, the main focus of research at product level has been failure/fault detection. Notable work has also been done in this domain. As an example (Harding, Shahbaz, & Kusiak, 2006), Lee, and Park (2001) used self-organizing maps (SOM), Sebzalli, and Wang (2001) applied a fuzzy clustering approach and Fountain, Dietterich, and Sudyka (2003) proposed Naïve Bayes classifiers for the purpose of fault detection. The use of resampling methods was also used in the work done by Cateni, Colla, and Vannucci (2014), which dealt with the study of two industry problems in the metal sector.

The natural development of failure detection is failure prediction at product level, which is also the focus of this article. Previous research in the direction of failure prediction at product level exists: Kusiak, and Kurasek (2001), Chen, Lee, Deng, and Liu (2007), Kim, Oh, Jung, and Kim (2018), Khan, Schiøler, Knudsen, and Kulahci (2015) and Khan, Schiøler, Kulahci, and Knudsen (2017). Moreover, Köksal, Batmaz, and Testik (2011) provide a detailed review of the use of data-mining methodologies in different segments of manufacturing.

4.2. Previous work in the area of imbalanced data classification

The area of imbalanced data classification has been extensively explored over the years and is evidenced by He and Ma (2013), the review work carried out by Sun, Wong, and Kamel (2009), and Haixiang et al. (2017). In the manufacturing

industry, the imbalance comes from the underrepresentation of faulty product data, which is not surprising, as the industry is subject to constant technological advancement.

Several basic strategies have been proposed to deal with imbalanced data classification, which can be divided into two main categories according to (Haixiang et al., 2017):

- **Preprocessing techniques**
  As the name indicates, these strategies tackle the imbalance problem using data preprocessing techniques, which can be feature engineering or resampling methods. Feature selection and extraction can alleviate the imbalance problem since, in some cases, the minority class samples can be discarded as noise. If the dimensionality is reduced and irrelevant features are removed, the risk of discarding minority class samples is reduced (Haixiang et al., 2017).
  Resampling methods can be over-sampling, under-sampling or of a hybrid nature. The idea behind these methods is to balance the two classes before applying the classifiers. One of the most popular methods in this category is the so-called Synthetic Minority Over-sampling Technique, or SMOTE (Domingos, 1999).

- **Cost-sensitive learning**
  This approach relies on the idea of assuming higher costs for the misclassification of minority class samples, and it can be incorporated at both the data level, for example, MetaCost (Domingos, 1999), or the algorithmic level (Haixiang et al., 2017). The basic idea is to define a cost matrix, as specified by Elkan (2001), and to modify the learning process by accepting costs. The cost matrix can be determined using process knowledge, or from data stream scenarios (Haixiang et al., 2017). Most of the time, the misclassification cost is unknown from the data or cannot be specified by an expert, and for this reason this type of learning is not particularly popular in the literature, as stated by Haixiang et al. (2017).
  It is important to mention that if cost-sensitive learning is not used for tackling the imbalance problem then this method is influenced by class imbalanced data (Liu, & Zhou, 2012).

The problem lies not only in the small sample size of the minority class. Other specific data problems (Napierała, & Stefanowski, 2012) might arise when dealing with imbalanced classification. Some of these issues, as specified by various research, can be classified into:

- **Small disjuncts**
  This problem arises when the minority class samples are decomposed into many sub-concepts with very few examples (Napierała, & Stefanowski, 2012; Japkowicz, 2003; Jo, & Japkowicz, 2004). This scenario produces difficulties in the learning process as a result of the lack of uniformity of the minority class and, especially, of the low number of samples in each sub-concept.

- **Class overlapping**
  In the case of class overlapping, which can also be considered class noise, there is a feature space region where the samples of the minority and majority classes contain a similar number of samples from each class (Lee, & Kim, 2018). In the case of a significant class overlap, the learning process from imbalanced data becomes a challenging problem (García, Mollineda, & Sánchez, 2008).

- **Noise**
  The noise present in the imbalanced data can come from different dimensions of data quality aspects, e.g. class noise (overlap), labelling errors, missing values or even attribute noise (Van Hulse, & Khoshgoftaar, 2009). For the manufacturing industry, besides the data quality aspects, the noise can come, for example, from sensors malfunctioning or other process-related problems. As in the case of class overlapping, the noise problem combined with the imbalanced data becomes a challenging problem to study (Van Hulse, & Khoshgoftaar, 2009; Napierała, Stefanowski, & Wilk, 2010).

This paper extends the work proposed by Khan, Schiøler, Zaki, and Kulahci (2018) and tries to alleviate some of the problems discussed, with a focus on the manufacturing industry. The machine learning challenge behind the industrial problem is the fact that the cost is fixed beforehand and that the problem is highly imbalanced with small disjuncts, class overlapping and noise.

5. Proposed classification strategy

In this section, we present the proposed classification strategy for dealing with the industrial problem. The method is formulated as a cost-sensitive learning strategy which also deals with the imbalance nature of the problem.

5.1. Classification strategy based on Voronoi diagram and genetic algorithm

The idea behind the classification strategy proposed by Khan, Schiøler, Zaki, and Kulahci (2018) is simple and consists of dividing the space into a number of pre-selected tiles, in which the tiles are assigned a class that is optimized using a stochastic optimization algorithm. Mathematically, this can be expressed as follows.

The feature space, X, is divided or tessellated into a pre-selected M number of tiles $\{T_1, \ldots, T_M\}$ where $T_j \subseteq X$, and these tiles are then grouped into two groups that define a binary partition, $\mathcal{B} = \{B_{pass}, B_{fail}\}$ of the X feature space. $\mathcal{B}$ is obtained by maximization/minimization of an appropriately designed fitness/objective function. As a result of the combinatorial nature of finding $\mathcal{B}$, stochastic optimization methods are appropriate for this task.

Practically, in the work carried out by Khan, Schiøler, Zaki, and Kulahci (2018), the above-mentioned steps are solved in the following way:

- The division of the feature space is done via Voronoi diagram (Lee, & Schachter, 1980), which requires seeds to be specified in advance.
- For the seed selection, the learning vector quantization (LVQ) (Kohonen, 1990; Nova, & Estévez, 2014) is used.
- The optimization is done using genetic algorithm (GA) (Goldberg, & Holland, 1988). The fitness/function used is described in detail below.

The fitness/objective function presented in Khan, Schiøler, Zaki, and Kulahci (2018) is built on the maximization of a ratio based on the difference between two confidence bounds.

For $i \in \{pass, fail\}$, let U_i and L_i be the upper and lower Clopper-Pearson confidence bounds (Clopper, & Pearson, 1934) of $\hat{P}(fail|B_{fail}) = \frac{X_{fail,B_{fail}}}{X_{.,B_{fail}}}$ and $\hat{P}(fail|B_{pass}) = \frac{X_{fail,B_{pass}}}{X_{.,B_{pass}}}$, then I statistic is defined as:

$$I = \frac{L_{fail} - U_{pass}}{(U_{pass} - L_{pass}) + (U_{fail} - L_{fail})} \tag{2}$$

This I statistic is built such that it renders statistically significant results by avoiding B_{pass} or B_{fail} being too small (Khan, Schiøler, Zaki, & Kulahci, 2018). Furthermore, the lower and upper confidence bounds are calculated using the beta transformation of the binomial distribution (Agresti, & Coull, 1998), which is built on the assumption that $X_{fail,B_{fail}}$ and $X_{fail,B_{pass}}$ are conditional binomials given $X_{.,B_{pass}}$ and $X_{.,B_{fail}}$.

5.2. Modified classification strategy as a cost-sensitive learning problem (VoronoiGA)

To solve the industrial problem, as mentioned above, we resorted to using the classification strategy from Section 5.1 with a few modifications that we considered appropriate for tackling the industrial problem. We denote the modified classification strategy problem as VoronoiGA further on in this article. The strength of this approach is that it can deal with class imbalance data problems and it is flexible enough to include an associated cost.

In terms of modifications, the method has been updated as follows:

- We changed the method of selecting the seed; LVQ is a supervised learning method and is impacted by highly imbalanced data, as stated by Grbovic, and Vucetic (2009). Even under these conditions, Nova, and Estévez (2014) mention that LVQ can still be trained without labels for clustering purposes, which is ultimately our purpose, as we are interested in finding the seeds that best represent the feature space. However, the industrial data received is also noisy, and it is known that LVQ algorithms are not robust for this kind of data, since sometimes the prototypes are trapped in positions where they are more harmful than helpful (Grbovic, & Vucetic, 2009). We therefore, decided to use k-means (Hartigan, & Wong, 1979) for the fail and pass data separately, thus having two different numbers of seeds to tune instead of one. We decided on the two parameters (pos.nr for fail data and neg.nr for pass data), as we did not want to miss important information regarding the distribution of the fail data points. Moreover, although k-means has some disadvantages, solely in terms of choosing the seeds we decided that this method would be sufficient.

- We made sure that the classification strategy would also be easy to implement for high dimensional data. Generating Voronoi diagram boundaries in high dimensions can become extremely computationally expensive as a result of the exponentially increased number of borders.
  There is a classic result, which links the Voronoi diagram with the k-nearest neighbor (k-NN), as specified by Mitchell (1997). Given the seeds, the Voronoi diagram is the same as the regions given by 1-NN, which makes this problem easily applicable in high dimensions. It is known that, as the number of dimensions increases, k-NN is subject to the curse of dimensionality (Pestov, 2013) and thus approximate methods can be used if necessary in order to avoid this possible problem (Indyk, & Motwani, 1998). Distance-wise, we decided to keep the Euclidean distance.

- We propose a new fitness/objective function for the GA algorithm that takes into consideration both the cost and the "quality". To do this, we first need to define what these measures are from a mathematical point of view and how they are connected to the industrial problem. The link along with the fitness/objective function is described into detail, below, in the remaining part of this section.

We start by defining that the fail class is the positive and the minority class, while the pass class is the negative and the majority class. The same convention is used in other studies, such as (Haixiang et al., 2017; Van Hulse, & Khoshgoftaar, 2009), with the confusion matrix defined as in Table 2 below.

Table 2
Confusion matrix

		Predicted class (status)		
		Positive (fail)	Negative (pass)	Total
Actual class (status)	Positive (fail)	True Positive (TP) $X_{fail,B_{fail}}$	False Negative (FN) $X_{fail,B_{pass}}$ *Type II error*	$X_{fail,.}$
	Negative (pass)	False Positive (FP) $X_{pass,B_{fail}}$ *Type I error*	True Negative (TN) $X_{pass,B_{pass}}$	$X_{pass,.}$
	Total	$X_{.,B_{fail}}$	$X_{.,B_{pass}}$	$X_{.,.}$

We are interested in detecting the FN (the faulty products sent to customers); in other words, the Type II error is more costly for the industrial manufacturing firm. The TP (faulty products classified as faulty) and FP (good products classified as faulty) also have associated costs; however, since they are detected and repaired internally, we decided not to include them as problematic quantities.

We define the cost from the industrial partner's perspective as the cost of checking all the products (cost of 1 per check), which in essence means that we classify everything as faulty products, namely, the quantity TP + FN + FP + TN. When a classifier is imposed, then the cost is reduced to C x FN + FP + TP, where C is a constant, which gives how much more expensive an FN is compared to FP and TP. We can then define the per unit cost as the ratio between the two quantities, as follows:

$$\text{Cost} = \frac{\text{C x FN} + \text{FP} + \text{TP}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{3}$$

For the measure of "quality", we consider it to be 1 – Sensitivity, where Sensitivity is defined as the ratio between the TP and the total number of positive samples, namely:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

The 1 – Sensitivity is a good "quality" measure for the industrial partner, as it can provide a quick overview of how many faulty products are potentially sent to the client in terms of percentage.

For imbalanced classification problems, there are other measures that can be used to see how well the classifier is performing. According to Haixiang et al. (2017), AUC/ROC, Accuracy, the Geometric Mean (GM) score and the F-score are some of the most popular methods to evaluate the performance of classifiers. It should be noted that even though AUC/ROC

is extremely popular, it has been questioned by Hand (2009). ROC is dependent on the cut-offs produced by the model, and these cut-offs are related to misclassification costs only when optimal thresholds are considered, thus making ROC an incoherent model (Haixiang et al., 2017). Counter-arguments regarding this interpretation also exist; see Ferri, Hernández-Orallo, and Flach (2011). Furthermore, Accuracy is biased towards the majority class, and thus it is not appropriate for imbalanced data; however, even in this case it is still used, as it is general and intuitive for classification.

Keeping the above arguments in mind, we decided to present the GM score and F-score, which in our view do not have any obvious disadvantages for the imbalanced data cases:

$$\text{GM score} = \sqrt{\text{Sensitivity} \cdot \text{Specificity}} \tag{5}$$

where Specificity is defined as TN/(TN + FP)) and the Sensitivity is defined above in (4). The GM score puts equal weight on both positive and negative classes.

$$\text{F-score} = (1+\beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}} \tag{6}$$

Precision is defined as TP/(TP + FP)) and Recall is in fact Sensitivity, as defined in (4). For the F-score, it does not take into consideration the TN quantity and, thus, it focuses on the detection of the performance of the positive class. In most of the applications $\beta = 1$ and then the score is called the F1 score.

To define the objective/fitness function for the GA algorithm, we decided to come up with a trade-off between cost (3) and Sensitivity (4). The trade-off is given by a user-defined parameter α and we write the problem as a minimization problem:

$$\text{Fitness function}(s, \alpha) = \alpha \cdot \text{Cost} + (1 - \alpha) \cdot (1 - \text{Sensitivity}) \tag{7}$$

Cost and Sensitivity are changed at each learning step based on the sub-selections (s) chosen by the GA algorithm. The trade-off methodology using the parameter α is a common way to deal with two measures, as, for example, in García, Triguero, Carmona, and Herrera (2012), which also deals with the problem of imbalanced classification.

6. Industrial problem framework

The following section is divided into two sub-sections that describe all the details behind solving the task proposed by our industrial partner. The first sub-section explains the chosen methodology, while the second sub-section contains the results (including the discussion and running time) obtained using the VoronoiGA strategy and other base classifiers on the problem.

6.1. Methodology for the industrial problem

Our proposed methodology is built on the assumption that an interdependence exists between subsequent stages of the manufacturing process. This assumption provides a reason to analyze collected data during earlier production stages to predict the final stage. The framework presented in Fig. 4 can be seen as a two-step learning strategy, where the first step deals with data-preparation matters such as feature engineering (including feature selection and extraction), and the second step is devoted to construction of the prediction model.

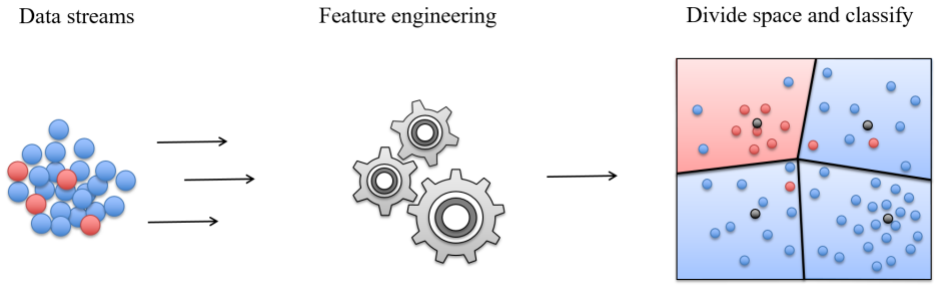


Fig. 4 Proposed two-step methodology

The idea of introducing a feature engineering step before building the final prediction model is a natural choice since the problem is high-dimensional and the classification methods are usually affected by the curse of dimensionality. For imbalanced learning problems, it is also a recommended step, as specified by Haixiang et al. (2017). The remainder of this

section presents the details of how the two-step methodology was implemented in practice.

6.1.1. *Programming language and running environment*

The chosen programming language is R and all the code was run on the DTU HPC Cluster.

6.1.2. *Data pre-processing and feature selection*

The first step was to make sure that the data did not contain any categorical data. We used one-hot-encoding to convert the categorical (assumed unordered) columns to multiple binary columns, where each column is made up of 1s and 0s indicating if the factor of the original column is present or not.

Given the industrial knowledge regarding the process, we created six new variables, which are mostly based on missing values, as presented in Table 3. These variables mainly expound the characteristics of each record in the data, such as when the first "NA" value occurred in the history of a particular record or the total number of missing values "NA" for a specific record. The next step we performed was to center and scale the data column-wise.

For the purpose of feature selection, we used a stochastic gradient boosting (Friedman, 2002) type of algorithm. The algorithm used was extreme gradient-boosting (XGBoost) (Chen & Guestrin, 2016), which is a tree-based supervised learning algorithm based on the principle of ensemble learning. We chose XGBoost to perform the feature selection based on its flexibility, because it can deal with missing data, as well as with highly imbalanced data using the scale_pos_weight parameter, (XGBoost Parameters, 2019). Furthermore, the implementation in R also offers the possibility to change the evaluation metric, which for this step was chosen as AUC (area under the curve).

Based on the industrial interpretation, we decided to run XGBoost twice, one time using as input the Sub Assembly 1 data with the final testing status
as output data, and the other time in the same manner but using the Sub Assembly 2 data. We had two purposes in mind when doing this procedure: one for dimensionality reduction and the other for feature selection.

The obtained gain percentages for each sub-assembly case were then used to obtain a space projection to a 2D space. Although extreme, we decided to project everything to a 2D space as a result of the industrial interpretation of the problem, namely, one axis for each sub-assembly. The projection was obtained by carrying out a linear combination between the feature data and the gain percentages for each feature. The samples containing "NA" were changed to 0s such that they were not influencing the final result. In other words, each sub-assembly data set was projected to a 1D with the help of XGBoost, Table 3.

Table 3
Feature engineering overview

Feature engineering procedure
***Input**: Data stream from sub-assemblies*
***Output**: Important features*
Merge data streams by ID;
For each** sub-assembly row **do
fst ← column # with first NA;
lst ← column # with last NA;
mx ← maximum value of row;
mn ← minimum value of row;
uniq ← # of unique values of row;
Naf ← # NA columns;
End
XGBoost (updated data for Sub Assembly 1)
XGBoost (updated data for Sub Assembly 2)

The data obtained from the feature engineering procedure was then used to assess the performance of the VoronoiGA strategy, along with other benchmark classifiers, in order to make a comparison.

6.1.3. *Classification models*

The modified strategy is a cost-sensitive learning strategy, and it is therefore important to define the cost. In the current industrial setting, a previous analysis revealed that the cost of a false negative (FN) is approximately 20 times higher than the cost of a false positive (FP), which means that C = 20 in equation (3). Since the cost depends on many factors, we also decided to consider a sensitivity analysis and to examine when the value of C is considerably higher, C = 100. For the α parameter from equation (7), we decided to investigate three different cases, meaning that we considered the values of alpha = 0.1, 0.5 and 0.9.

To assess the generalization performance of the VoronoiGA strategy under different parameter values of C and α, we decided to use a nested cross-validation procedure. We preferred a nested cross-validation procedure, as opposed to standard

cross-validation, since the test data for each of the outer cross-validation loops had not been used to optimize the model performance before. If standard cross-validation is used, the general impression is that the results are too optimistic (biased) for the generalization performance. This is mentioned by other research, for example, Cawley, and Talbot (2010), Krstajic, Buturovic, Leahy, and Thomas (2014) or, more recently, Wainer, and Cawley (2018). The downside of the nested cross-validation procedure is the added computational cost, which is higher than in the case of standard cross-validation. For our work, we used 5 folds for the outer loop and 5 folds for the inner loop to fine-tune the hyperparameters. To assess the uncertainty, we repeated the nested cross-validation procedure randomly 50 times. The mean and standard deviation of the results were computed and presented in the form of mean ± standard deviation, for which we also presented the running time. In this case, by running time, we mean the average running time over the 50 randomly repeated nested cross-validations.

There are different ways to tune the hyperparameters. The most common approaches are grid search, random search and Bayesian optimization. We decided to use Bayesian optimization (Snoek, Larochelle, & Adams, 2012; Frazier, 2018). In our work, we used Bayesian optimization on the inner loop of the nested cross-validation using the trade-off measure (equation (7)) average over the inner folds. To find the optimal hyperparameters, Bayesian optimization needs a user-defined number of runs. Each run searches the space in an optimized manner, with the final purpose of finding the maximum trade-off measure average. It should be noted that, as the number of hyperparameters increases, an exponential number of runs should be considered to span the same space. We opted for 20 runs, in order to make sure that we have enough runs to span the hyperparameter space for all the considered classifiers. Furthermore, we fine-tuned the hyperparameter intervals in advance before running the Bayesian optimization.

VoronoiGA was compared with different base classifiers in order to evaluate the strengths and weaknesses of the strategy when applied to the industrial problem. There is a high number of proposed algorithms in the imbalanced learning literature, and therefore we decided to compare the strategy with the most common base classifiers from previous studies, as provided by Haixiang et al. (2017). Since the problem is highly imbalanced, most of the classifiers were also run with a common data resampling strategy, which was chosen as SMOTE (Haixiang et al., 2017). We also decided to run k-NN without a data resampling strategy since the VoronoiGA strategy is based on the 1-NN classifier. According to Haixiang et al. (2017), SVM is the most popular base classifier, while previous work showed that SVM is more robust to imbalanced data than other classifiers (Yu et al., 2015). Hence, we ran SVM with different sampling strategies, as presented in Table 4. Another obvious choice of algorithm is XGBoost, since we used it for feature selection because of its flexibility to class imbalance. Since the implementation in R also permits us to change the evaluation metric, we used the default AUC and also the one based on the trade-off measure (equation (7)) in order to see how sensitive XGBoost is to different evaluation metrics.

We decided to run VoronoiGA with the two types of fitness/objective function (equations (2) and (7)) to check if the strategy performs differently in these cases. All classifiers that were used, along with the parameters, are presented in Table 4.

Table 4

Parameters used for the algorithms, along with the corresponding abbreviations

Algorithm/Classifier	Abbreviation Algorithm/Classifier	Parameters
Voronoi with Genetic Algorithm	VoronoiGA	GA fitness function uses the trade-off between cost and Sensitivity, i.e. it is minimizing the cost and maximizing the Sensitivity and pos.nr = [80, 105], neg.nr = [45, 60]
Voronoi with Genetic Algorithm	VoronoiGAIstat	GA fitness function uses the I statistic function and pos.nr = [80, 105], neg.nr = [45, 60]
Extreme Gradient Boosting (Chen & Guestrin, 2016)	XGBoost	eval.metric of XGBoost uses the trade-off between cost and Sensitivity function, i.e. it is minimizing the cost and maximizing the Sensitivity max.depth = [3, 10], min_child_weight = [1, 40], subsample = [0.6, 0.9], scale_pos_weight = 72, eta=[0.1, 0.3], gamma = [0, 0.2], colsample_bytree = [0.5, 0.8], max_delta_step=[1, 10]
Extreme Gradient Boosting (Chen & Guestrin, 2016)	XGBoostAuc	eval.metric is maximizing the accuracy, max.depth = [3, 10], min_child_weight = [1, 40], subsample = [0.6, 0.9], scale_pos_weight = 72, eta=[0.1,0.3], gamma = [0, 0.2], colsample_bytree = [0.5, 0.8], max_delta_step=[1, 10]
Support Vector Machine (Vapnik, 1998) with Under-sampling	svmUNDER	Radial basis function kernel, gamma = $[2^{-2}, 2^{10}]$, cost = $[2^{-2}, 2^{10}]$ and N (sample size) = 2 times number of positive classes
Support Vector Machine (Vapnik, 1998) with Synthetic Minority Over-sampling Technique	svmSMOTE	Radial basis function kernel, gamma = $[2^{-2}, 2^{10}]$, cost = $[2^{-2}, 2^{10}]$ and SMOTE k = 5, percentage over = 100, percentage under=200 (this creates 50% - 50% distribution among classes)
Support Vector Machine (Vapnik, 1998) with Under-sampling and Over-sampling	svmBOTH	Radial basis function kernel, gamma = $[2^{-2}, 2^{10}]$, cost = $[2^{-2}, 2^{10}]$ and p = 0.5 (probability of resampling from the positive class) and N (sample size) = 2 times number of positive classes
Random Forest (Friedman, Hastie, & Tibshirani, 2001) with Synthetic Minority Over-sampling Technique	rfSMOTE	nr.trees = [100, 500] and SMOTE k = 5, percentage over = 100, percentage under=200 (this creates 50% - 50% distribution among classes)

Artificial Neural Networks (Mitchell, 1997) with Synthetic Minority Over-sampling Technique	nnSMOTE	One single layer with neurons = [1, 5] and SMOTE k = 5, percentage over = 100, percentage under=200 (this creates 50% - 50% distribution among classes)
Naive Bayes (Mitchell, 1997) with Synthetic Minority Over-sampling Technique	nbSMOTE	No parameters to tune and SMOTE k = 5, percentage over = 100, percentage under=200 (this creates 50% - 50% distribution among classes)
Logistic regression (Friedman, Hastie, & Tibshirani, 2001) with Synthetic Minority Over-sampling Technique	logregSMOTE	No parameters to tune and SMOTE k = 5, percentage over = 100, percentage under=200 (this creates 50% - 50% distribution among classes)
k-Nearest Neighbors (Mitchell, 1997) with Synthetic Minority Over-sampling Technique	knnSMOTE	k = [1, 20] and SMOTE k = 5, percentage over = 100, percentage under=200 (this creates 50%, 50% distribution among classes)
k-Nearest Neighbors (Mitchell, 1997)	kNN	k = [1, 20]

From an industrial point of view, when comparing the classifiers, individual ranking of cost and Sensitivity are not meaningful, since the industrial partner is interested in finding the classifier that gives the minimum cost with the maximum Sensitivity. Thus, we are interested in ranking the pair of the two measures, which we believe is cumbersome to check from a statistical point of view. However, from a research comparison perspective, it is important to check if the results obtained among the different classifiers are statistically different. Therefore, to assess the generalization performance of the chosen classifiers, we want to make sure that there is a significant difference among the groups in terms of cost and Sensitivity separately. For this, we employ a non-parametric test for multiple comparisons, namely, the Friedman test (García, Fernández, Luengo, & Herrera, 2009; García, Fernández, Luengo, & Herrera, 2010), which tests whether the average ranks of the classifiers over different scenarios are significantly different. In other words, we are checking these two hypotheses:

H_0 : There is no difference in the mean of the classifiers' average ranks over all the scenarios.

H_1 : There is a difference in the mean of the classifiers' average ranks over all the scenarios.

The ranking is achieved by assigning a position to each classifier depending on the performance for each scenario. The classifier that achieves the best cost/Sensitivity on a specific scenario gets the first ranking (as 1); then, the classifier with the second-best cost/Sensitivity is assigned the second ranking (as 2), and so on. This procedure is carried out for all the existing scenarios and classifiers and, finally, the average ranking is returned as the mean of all rankings (García, Triguero, Carmona, & Herrera, 2012). It is also common in the machine learning literature to find which classifiers are distinctive among the comparisons. For this, the Holm post hoc test (García, Fernández, Luengo, & Herrera, 2010) is usually used. By computing the adjusted p-value (APV) associated with each comparison, the post hoc procedure tests if a means comparison hypothesis can be rejected at a specified level of significance. Furthermore, the computed APV for each comparison represents the lowest significance level that results in a hypothesis rejection. For the purpose of this article, we consider that a significance level of 0.05 is sufficient for both tests.

Since our ultimate goal for the industrial problem is not necessarily the comparison between VoronoiGA and the other classifiers in terms of separate cost/Sensitivity ranking, we decided not to employ the Holm post hoc test but only the Friedman test. For a fairer comparison of the VoronoiGA strategy with the base classifiers, we used benchmark data sets with other measures rather than the industrial problem.

6.2. Results and comparison study for the industrial problem

The results obtained following the methodology presented above are presented in this section. The results are divided into two sub-sections, with one presenting the feature engineering results and the other presenting the VoronoiGA results, along with the base classifier comparison.

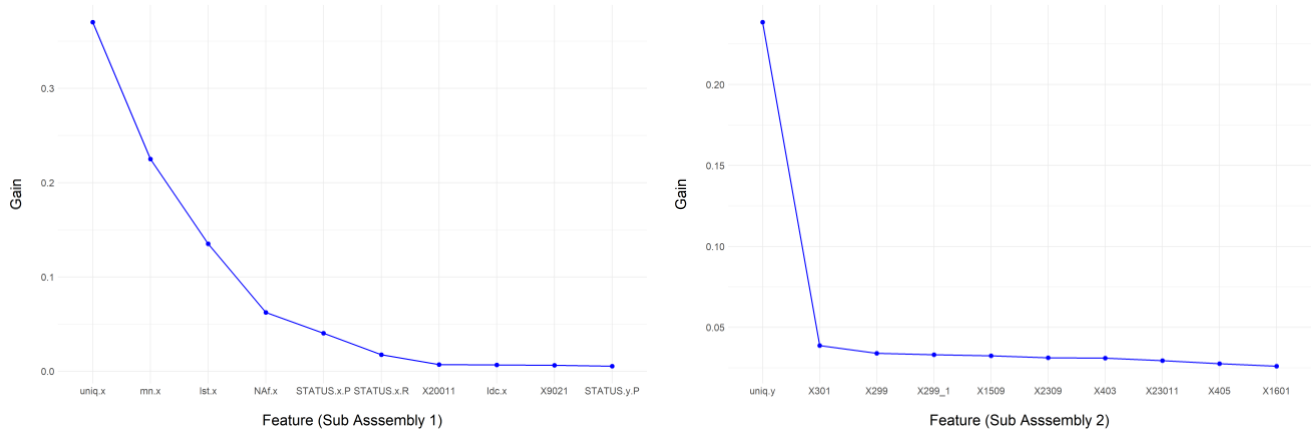
6.2.1. Results for feature engineering and data presentation

We first applied the XGBoost algorithm for dimensionality reduction and feature selection. In Table 5 the number of selected features is displayed after applying XGBoost using as input the Sub Assembly 1 and 2 data sets and as output the final testing status. We can observe that XGBoost decreases the feature space considerably, and therefore it is more accessible to work with the data in this form.

Table 5
Characteristics of sub-assembly lines

	Initial dimensions	Selected features
Sub Assembly 1 (control card)	5,021	71
Sub Assembly 2 (power card)	5,072	37

In Fig. 5, the gains obtained for both cases are plotted using the first 10 selected features from Table 5. As can be observed, for the case (a) of Sub Assembly 1, the first four features, which account for most of the gain percentages, come from the six newly engineered variables. The rest of the features are specific process features, which are not interesting for the reader and, thus, will not be explained in detail. The number of unique values in a row account for most of the gain percentage of approximately 37% in this case. For the case (b) of Sub Assembly 2, we can observe that again the number of unique row values accounts for the highest gain percentage, namely, 24 %. The remaining features are also in this case process-specific features and will not be explained in detail.



(a) (b)

Fig. 5 Gain for the features of Sub Assembly 1 and 2

The resulting 2D space from the feature engineering final step (the linear combination between the samples and the gains) is presented in Fig. 6. As can be deducted from the figure, the data is multi-modal, noisy and the classes overlap considerably. Furthermore, there is a space region where the two classes are separated and thus the classifiers can benefit from it.

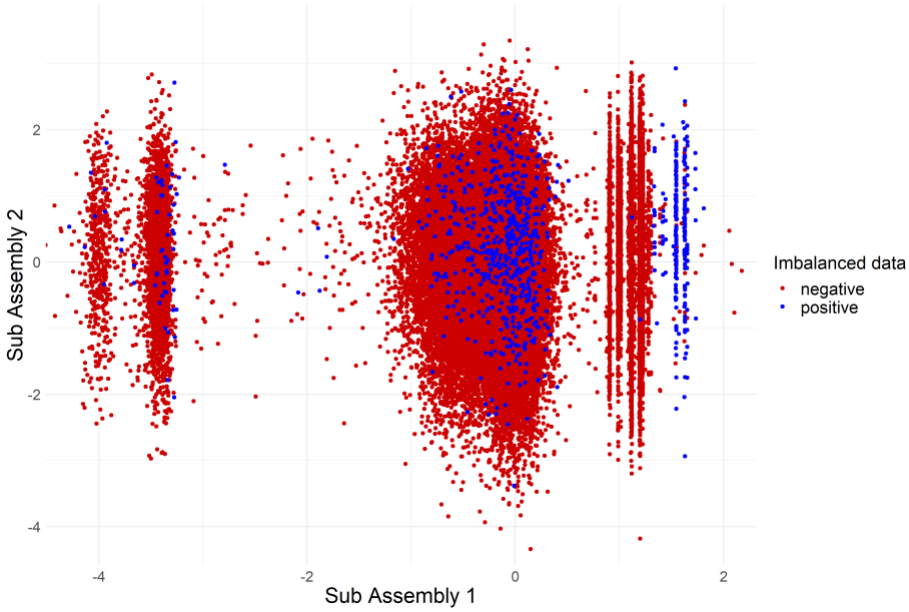
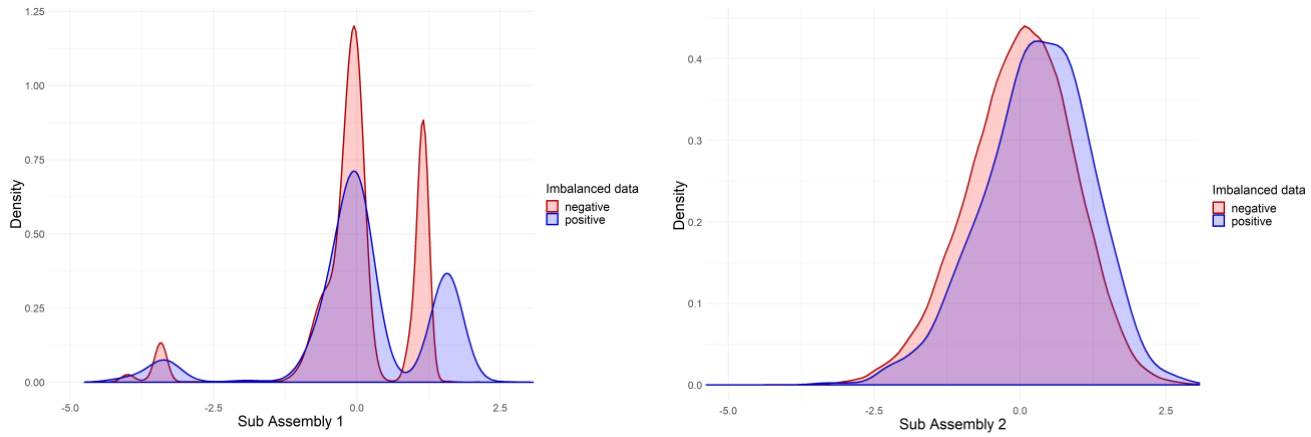


Fig. 6 Sub Assembly 1 versus Sub Assembly 2 processed data

In order to obtain a better overview of the class separation possibilities, we also decided to present the densities on both axes. As can be seen in Fig. 7, there is a class separation on both axes, which classifiers can benefit from. On the Sub Assembly 1 axis, case (a) of Fig. 7, there are three different modes and one mode yields a better separation. The same cannot be said for the Sub Assembly 2 axis, case (b) of Fig. 7, where the two classes overlap.



(a) (b)

Fig. 7 Densities of Sub Assembly 1 and 2.

6.2.2. Results and comparison study for the industrial problem

This section presents the results that were obtained for the VoronoiGA strategy, as well as the comparison with the base classifiers discussed in the methodology section.

Since the resulting problem is in a 2D space, it is easy to show how the VoronoiGA is performing visually. We chose, for illustration purposes, an example of pos.nr = 100 and neg.nr = 100 for $\alpha = 0.1$ and $C = 20$, in order to obtain a good overview of how k-means selects the seeds and also how the classification is performed. In sub-figure (a) Fig. 8, we can see the class distribution based on where the original seed was selected and, as can be observed, the tiles are smaller in the overlapping region since both seeds were extracted from the same region. In sub-figure (b) Fig. 8, it can be observed that VoronoiGA is correctly identifying the negative and positive regions when the regions are separable, whereas the overlapping regions are mostly classified as positive.

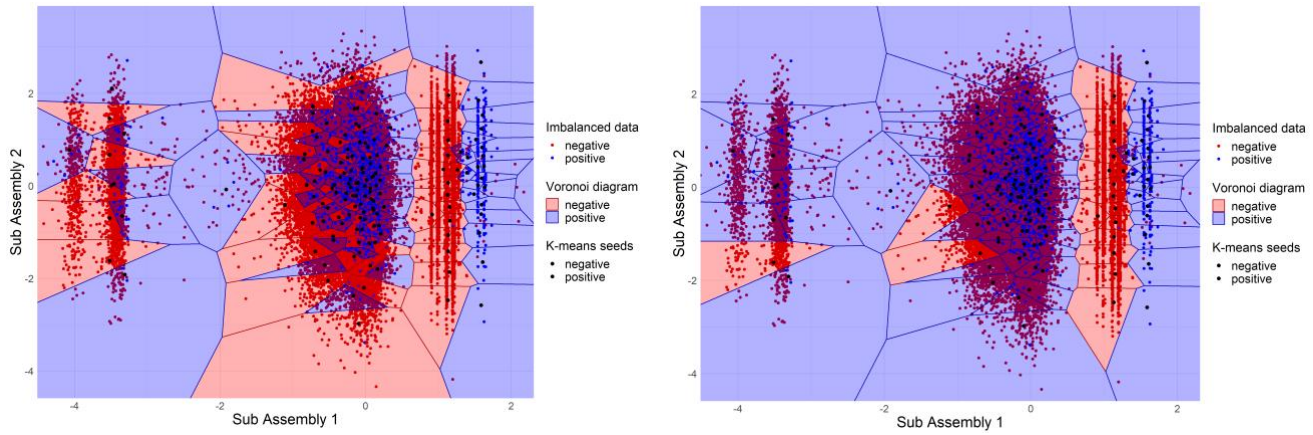


Fig. 8 Division of the Sub Assembly 1 versus Sub Assembly 2 space using pos.nr = 100 and neg.nr = 100; sub-figure (a) represents the initial colouring, namely, the colour (class) of the initial seed, while sub-figure (b) is coloured using the algorithm VoronoiGA with $\alpha = 0.1$ and $C = 20$

The cost and Sensitivity results from running all the classifiers on all the possible scenarios are displayed in Table 6. We represented in bold and underlined the minimum of cost and the maximum of Sensitivity.

As shown in Table 6, we can observe visually that the cost and Sensitivity results vary for different classifiers. When employing the Friedman test for the cost results, we obtained a value of 25.45 for the statistic with a p-value of 0.013, which tells us there is a difference among the mean of the classifiers' average ranks at a significance level of 0.05. For the Sensitivity results, we obtained a value of 64.02 for the statistic with an associated p-value of $4.13 \cdot 10^{-9}$, which is also significant at the level of 0.05. In both cases, there are 12 degrees of freedom for the test statistic's approximate chi-squared distribution, since there are 13 different classifiers in Table 6. This leads to the conclusion that the average ranks of cost and Sensitivity are statistically different at a level of 0.05 for the compared classifiers using the six different scenarios.

Table 6
Cost and Sensitivity results for the industrial data set; the minimum of cost and maximum of Sensitivity are displayed in bold and underlined

Score	Type	VoronoiGA	VoronoiGAIstat	XGBoost	XGBoostAuc	svmUNDER	svmSMOTE	svmBOTH	rfSMOTE	nnSMOTE	nbSMOTE	logregSMOTE	knnSMOTE	kNN
Cost	C=20, α=0.1	.7531 ± .0054	.7220 ± .0239	.3393 ± .0032	.3391 ± .0025	.4716 ± .0228	.3850 ± .0152	.3676 ± .0176	.3890 ± .0043	.4120 ± .0375	.5264 ± .0388	.5694 ± .0017	.4335 ± .0046	**.2153 ± .0012**
	C=20, α=0.5	.5081 ± .0079	.7108 ± .0304	.3375 ± .0038	.3370 ± .0039	.4018 ± .0257	.3830 ± .0129	.3287 ± .0159	.3903 ± .0043	.4032 ± .0190	.5179 ± .0373	.5697 ± .0027	.4334 ± .0044	**.2151 ± .0007**
	C=20, α=0.9	.2187 ± .0017	.7051 ± .0358	.2130 ± .0013	**.2127 ± .0010**	.3802 ± .0207	.3842 ± .0133	.2910 ± .0147	.3889 ± .0046	.2994 ± .0590	.5310 ± .0459	.5704 ± .0029	.4192 ± .0075	.2151 ± .0007
Sensitivity	C=20, α=0.1	**.9822 ± .0049**	.9519 ± .0234	.5566 ± .0083	.5570 ± .0093	.6296 ± .0359	.5948 ± .0156	.5295 ± .0217	.5886 ± .0135	.6506 ± .0466	.5444 ± .0347	.5786 ± .0048	.6390 ± .0127	.2318 ± .0024
	C=20, α=0.5	.7438 ± .0149	**.9407 ± .0315**	.5494 ± .0111	.5473 ± .0122	.6092 ± .0212	.5929 ± .0230	.5056 ± .0227	.5854 ± .0128	.6651 ± .0290	.5369 ± .0371	.5788 ± .0045	.6409 ± .0114	.2315 ± .0027
	C=20, α=0.9	.2906 ± .0072	**.9390 ± .0346**	.2829 ± .0049	.2818 ± .0041	.5824 ± .0283	.4730 ± .0427	.1694 ± .0943	.5895 ± .0141	.3794 ± .0747	.5450 ± .0454	.5788 ± .0054	.5979 ± .0205	.2315 ± .0028
Cost	C=100, α=0.1	**.7737 ± .0060**	.7743 ± .0141	.8275 ± .0136	.8314 ± .0119	.8849 ± .0275	.8287 ± .0178	.8868 ± .0236	.8452 ± .0175	.7945 ± .0326	1.0271 ± .0117	1.0330 ± .0043	.8270 ± .0154	1.0591 ± .0038
	C=100, α=0.5	**.7580 ± .0128**	.7720 ± .0131	.8322 ± .0117	.8319 ± .0102	.8617 ± .0208	.8301 ± .0165	.8759 ± .0220	.8434 ± .0169	.7831 ± .0251	1.0282 ± .0117	1.0336 ± .0053	.8290 ± .0151	1.0605 ± .0049
	C=100, α=0.9	.7763 ± .0147	**.7723 ± .0132**	.8284 ± .0121	.8331 ± .0105	.8381 ± .0170	.8293 ± .0200	.8660 ± .0177	.8412 ± .0164	.7795 ± .0215	1.0292 ± .0129	1.0329 ± .0051	.8309 ± .0167	1.0593 ± .0039
Sensitivity	C=100, α=0.1	**.9831 ± .0040**	.9449 ± .0329	.5565 ± .0105	.5530 ± .0096	.6180 ± .0321	.5958 ± .0166	.5203 ± .0248	.5858 ± .0132	.6383 ± .0572	.5463 ± .0350	.5784 ± .0036	.6414 ± .0124	.2317 ± .0027
	C=100, α=0.5	.9158 ± .0118	**.9375 ± .0338**	.5517 ± .0090	.5519 ± .0081	.6207 ± .0239	.5962 ± .0163	.5177 ± .0274	.5867 ± .0130	.6604 ± .0428	.5392 ± .0390	.5780 ± .0041	.6404 ± .0127	.2307 ± .0034
	C=100, α=0.9	.8045 ± .0148	**.9425 ± .0287**	.5539 ± .0089	.5493 ± .0092	.6063 ± .0239	.5915 ± .0172	.5156 ± .0240	.5887 ± .0126	.6579 ± .0356	.5424 ± .0340	.5787 ± .0042	.6378 ± .0131	.2316 ± .0028

Before discussing the results, we will present what we expect to see from the classification naïve models in terms of cost and Sensitivity for both cases of C = 20 and C = 100. There are three different naïve model scenarios: 1) when the model is classifying everything as positive; 2) when the model is classifying everything as negative; and 3) when the model is not misclassifying any observations.

1) If the model is classifying everything as positive, then FN = 0 and TN = 0, which means that Sensitivity = 1 and cost = 1 for both C = 20 and C = 100. This is in fact the scenario where all the products are checked one by one.

2) If the model is classifying everything as negative, then FP = 0 and TP = 0, which means that Sensitivity = 0 and cost = 0.27 for C = 20 and Sensitivity = 0 and cost = 1.37 for C = 100.

3) If the model is not misclassifying observations then FN = 0 and FP = 0, which means that the Sensitivity = 1 and cost = 0.014 for both C = 20 and C = 100.

In the best-case scenario, that is, no observations are misclassified by the classifier, a reduction in cost of 98.6% can be obtained, with no bad products sent to the customer (Sensitivity = 1). This means that this is the minimum cost threshold, which can be obtained using any classifier. On the other hand, if the model is classifying everything as negative (good products) then a reduction of 73% (C = 20) can still be obtained at the expense of all faulty products being sent to the client. However, this is not the case, if C = 100 and if faulty products sent to the client are penalized more. In this case, it is best to check all the products one by one, as the cost will be smaller, with possibly no bad products sent to the customer. In general, there is a trade-off between cost and Sensitivity and it is the responsibility of the industrial partner to choose this optimal business trade-off.

Returning to the results from Table 6, as it can be observed, the results are different for the cases of C = 20 and C =100 for most of the classifiers; thus we will discuss them in detail separately.

For the C = 20 case, we can observe that VoronoiGA implemented with the trade-off function is highly sensitive to the parameter α, as opposed to VoronoiGA implemented with the I statistic, where the results are still varying but not to the same extent. This result indicates that the chosen fitness/objective is highly important for the strategy and also shows flexibility. The same cannot be said for XGBoost, where we also changed the evaluation metric and the obtained results are very similar. As a general rule, most of the classifiers are sensitive to the parameter α, as a result of the hyperparameter tunning in the inner loop of the nested cross-validation, though none of the classifiers is flexible enough to get to high values of Sensitivity. From the industrial perspective, the VoronoiGA strategy has a clear advantage over the other classifiers, as the industrial partner can fine-tune the method at the desired level of cost and Sensitivity. For example, using the VoronoiGA strategy, a Sensitivity of 98% will lead to a cost reduction of 25% from the base case where all the products are checked one by one.

In the case of C = 100, the results obtained for all classifiers are not sensitive to the α parameter, as in the case of C = 20. The reason behind this is the extra weight given by the C parameter in the trade-off function, which affects the final cost. Even under these conditions, we can still observe that the VoronoiGA strategy implemented with the trade-off function is still reacting the most of all the classifiers to the α parameter in terms of Sensitivity. In this case, it is easy to observe that VoronoiGA performs the best, since it obtains the minimum cost with the highest Sensitivity. Since C is high, some classifiers even exceed the cost of 100%, as in the case of naïve Bayes with SMOTE, logistic regression with SMOTE and k-NN.
The remarkable result for the VoronoiGA strategy is that, even in this case, when C is much higher, a similar result to the case of C = 20 can be obtained. The reduction of cost from the same base case is decreased by 2%, namely, 23%, and Sensitivity is still at the 98% level.

Finally, we present in Table 7 the running times for all the classifiers using the HPC cluster. We have underlined and presented in bold the fastest-running classifiers. Since the naïve Bayes and logistic regression algorithms do not have parameters to tune, it is obvious that they are performing the fastest of all the classifiers.

In terms of computational cost, it is expected for the VoronoiGA strategy to have a higher computational cost because of the combinatorial nature of the strategy. Even in these circumstances, it can be observed that the strategy implemented with the trade-off measure runs faster than XGBoost and is comparable in terms of running time with the artificial neural networks model. The fitness/objective function of the VoronoiGA strategy is important for the running time, as seen in Table 7, since the GA algorithm is evaluating the function multiple times until convergence.

For the industrial set-up, the running time is not necessarily important, as the classifier is built offline and only the final model is deemed to be used online when predicting the status of the products sent to the client.

Table 7
Running time in seconds for all the classifiers; the fastest-running classifiers for each scenario are presented in bold and underlined

Type	VoronoiGA	VoronoiGAIstat	XGBoost	XGBoostAuc	svmUNDER	svmSMOTE	svmBOTH	rfSMOTE	nnSMOTE	nbSMOTE	logregSMOTE	knnSMOTE	kNN
C=20, α=0.1	3653 ± 557	7304 ± 283	9277 ± 690	5183 ± 229	648 ± 52	1551 ± 67	594 ± 48	549 ± 42	3229 ± 344	14.02 ± .48	**2.29 ± .24**	400 ± 29	3509 ± 87
C=20, α=0.5	3943 ± 346	8047 ± 927	9128 ± 586	5138 ± 148	632 ± 55	1273 ± 64	643 ± 51	555 ± 54	3223 ± 302	11.29 ± 1.75	**2.38 ± .28**	391 ± 24	3489 ± 59
C=20, α=0.9	3871 ± 429	7409 ± 223	9843 ± 408	5619 ± 153	541 ± 22	1068 ± 41	522 ± 42	559 ± 55	1926 ± 230	**1.35 ± .41**	2.26 ± .20	369 ± 20	3354 ± 50
C=100, α=0.1	3235 ± 84	7416 ± 102	9833 ± 618	5061 ± 109	558 ± 28	1295 ± 106	619 ± 48	563 ± 40	2821 ± 278	**1.61 ± 1.14**	2.30 ± .22	404 ± 26	3086 ± 161
C=100, α=0.5	3582 ± 476	7311 ± 136	9782 ± 709	4781 ± 135	570 ± 41	1390 ± 151	604 ± 57	561 ± 35	2799 ± 221	11.66 ± 1.89	**2.34 ± .26**	402 ± 25	3395 ± 152
C=100, α=0.9	3094 ± 76	7311 ± 148	9327 ± 649	5099 ± 430	674 ± 45	1549 ± 88	609 ± 62	560 ± 58	2765 ± 180	14.32 ± .08	**2.35 ± .24**	398 ± 25	3445 ± 86

7. Experimental framework

We wanted to demonstrate that the modified VoronoiGA strategy can also be applied for imbalanced data in cases when the cost is not available, as this is one of the downsides of the cost-sensitive learning methods (Haixiang et al., 2017). For this study, we chose 25 real-world data sets, Table 9, with different imbalance ratios from the KEEL repository (Alcalá-Fdez et

al., 2011). In terms of the VoronoiGA strategy, a different fitness/objective function was used to account for the missing cost. Furthermore, in order to assess the performance of the VoronoiGA strategy, we also performed a comparison study with the same base classifiers as those used for the industrial set-up but modified accordingly.

This section contains all the details regarding the comparison study and is divided into two sub-sections. The first sub-section talks about the applied methodology details, while the other sub-section presents the results and discusses them.

7.1. Methodology for the benchmark study

The methodology for the comparison study on the benchmark data is very similar to the one followed for the industrial problem set-up, Section 6.1.3. There are a few changes in terms of the classifiers' parameters, which are displayed in Table 8. The changes are mostly in the hyperparameter space used by the Bayesian optimization since we kept the 20 runs. We also decided to change the resampling parameters where we considered it necessary.

Table 8
Parameters used for the classifiers, along with the corresponding abbreviations

Abbreviation Algorithm/Classifier	Parameters
VoronoiGA	GA fitness/objective function is maximizing the GM score, poss.nr and neg.nr have been adjusted differently for each data set (25 entries in every vector), i.e. pos.nr.min = [1, 2, 15, 5, 1, 1, 1, 1, 1, 2, 1, 1, 5, 1, 1, 10, 1, 1, 1, 5, 5, 10, 1, 1, 1] pos.nr.max = [25, 10, 25, 15, 10, 3, 2, 25, 25, 3, 25, 25, 10, 5, 3, 25, 5, 5, 3, 20, 10, 25, 25, 25, 25] neg.nr.min = [1, 20, 20, 15, 15, 30, 30, 10, 25, 30, 25, 1, 5, 1, 1, 20, 90, 1, 20, 10, 1, 8, 25, 10, 25] neg.nr.max = [5, 30, 50, 20, 40, 35, 40, 50, 100, 40, 100, 100, 10, 10, 6, 50, 120, 5, 35, 40, 10, 100, 100, 40, 100]
XGBoost	the eval.metric of XGBoost is maximizing the GM score, max.depth = [3, 10], min_child_weight = [1, 40], subsample = [0.6, 0.9], scale_pos_weight = 72, eta=[0.1, 0.3], gamma = [0, 0.2], colsample_bytree = [0.5, 0.8], max_delta_step=[1, 10]
svmUNDER	Radial basis function kernel, gamma = $[2^{-10}, 2^{3}]$, cost = $[2^{-10}, 2^{3}]$ and N (sample size) = 2 times number of positive classes
svmSMOTE	Radial basis function kernel, gamma = $[2^{-10}, 2^{3}]$, cost = $[2^{-10}, 2^{3}]$ and SMOTE k = 5, percentage over = 500, percentage under=100
svmBOTH	Radial basis function kernel, gamma = $[2^{-10}, 2^{3}]$, cost = $[2^{-10}, 2^{3}]$ and p = 0.8 (probability of resampling from the positive class) and N (sample size) = 2 times number of positive classes
rfSMOTE	nr.trees = [100, 500] and SMOTE k = 5, percentage over = 500, percentage under = 100
nnSMOTE	One single layer with neurons = [1, 5] and SMOTE k = 5, percentage over = 500, percentage under = 100
nbSMOTE	No parameters to tune and SMOTE k = 5, percentage over = 500, percentage under = 100
logregSMOTE	No parameters to tune and SMOTE k = 5, percentage over = 500, percentage under = 100
knnSMOTE	k = [1, 20] and SMOTE k = 5, percentage over = 500, percentage under = 100
kNN	k = [1, 20]

Another notable change is in the nested cross-validation procedure, as we opted to use 3 inner folds as opposed to 5 inner folds. We made this change in order to ensure that we had enough positive and negative classes in the validation data set. The results are again presented in the form of mean ± standard deviation for 50 randomly nested cross-validations, along with the running time, computed in the same manner as in Section 6.1.3.

However, the major change comes from assessing the performance of the methods. In this case, since it is an imbalanced classification problem, we used a different metric to see how the classifiers are performing. In Section 5.2, we present the GM and F1 scores and, since GM takes into consideration both the TP and TN, we decided to focus on GM. Thus, we decided to maximize GM when tuning the hyperparameters in the nested cross-validation and also to maximize the GM score for the VoronoiGA and XGBoost algorithms. For a fair comparison, we also decided to report the F1 scores that were obtained, in order to see how all the methods were performing in this case too. For the comparison, we again used the Friedman test and, in this case, we also reported the adjusted p-values with the help of the Holm post hoc test. In addition, also in this case, we considered that a significance level of 0.05 is sufficient for both tests.

The selected 25 KEEL data sets are presented in ascending order of the imbalance ratio, as can be seen in Table 9. For the abalone data sets, we also performed a one-hot-encoding to make sure that the data sets do not contain nominal data. We did not center or standardize the data.

Table 9
Description of the KEEL real-world data sets

Data sets	Number of features	Number of observations in the positive class	Number of observations in the negative class	Total number of observations	Imbalance ratio	Attributes (Real/Integer/Nominal)
wisconsin	9	239	444	683	1.86	(0/9/0)
haberman	3	81	225	306	2.78	(0/3/0)
new-thyroid1	5	35	180	215	5.14	(4/1/0)
glass6	9	29	185	214	6.38	(9/0/0)
ecoli-0-4-6_vs_5	6	20	183	203	9.15	(6/0/0)
glass-0-1-6_vs_2	9	17	175	192	10.29	(9/0/0)
glass2	9	17	197	214	11.59	(9/0/0)
shuttle-c0-vs-c4	9	123	1706	1829	13.87	(0/9/0)
glass4	9	13	201	214	15.46	(9/0/0)
glass-0-1-6_vs_5	9	9	175	184	19.44	(9/0/0)
shuttle-c2-vs-c4	9	6	123	129	20.5	(0/9/0)
shuttle-6_vs_2-3	9	10	220	230	22	(0/9/0)
yeast-2_vs_8	8	20	462	482	23.1	(8/0/0)
yeast4	8	51	1433	1484	28.1	(8/0/0)
yeast-1-2-8-9_vs_7	8	30	917	947	30.57	(8/0/0)
yeast5	8	44	1440	1484	32.73	(8/0/0)
ecoli-0-1-3-7_vs_2-6	7	7	274	281	39.14	(7/0/0)
yeast6	8	35	1449	1484	41.4	(8/0/0)
winequality-white-3_vs_7	11	20	880	900	44	(11/0/0)
abalone-19_vs_10-11-12-13	8	32	1590	1622	49.69	(7/0/1)
winequality-white-3-9_vs_5	11	25	1457	1482	58.28	(11/0/0)
shuttle-2_vs_5	9	49	3267	3316	66.67	(0/9/0)
abalone-20_vs_8-9-10	8	26	1890	1916	72.69	(7/0/1)
poker-8-9_vs_5	10	25	2050	2075	82	(0/10/0)
abalone19	8	32	4142	4174	129.44	(7/0/1)

7.2. Results for the benchmark study

In this section we present the results obtained when applying the classifiers on the KEEL data sets.

The GM scores obtained are presented in Table 10. As can be observed, the classifiers perform differently on the data sets. When applying k-NN on some data sets, the classifier was only predicting as the negative class, which made the GM score zero. In order to see if the results are statistically different in terms of ranking, we applied the Friedman test. For the GM scores, the degrees of freedom are 10 and we obtained a value of 83.38 for the statistic, with a p-value of $1.09 \cdot 10^{-13}$, which indicates that there is a difference among the mean of the classifiers' average ranks at a significance level of 0.05.

Table 10
GM score for the selected KEEL data sets; the maximum GM scores for each data set are displayed in bold and underlined

Data sets	VoronoiGA	XGBoost	svmUNDER	svmSMOTE	svmBOTH	rfSMOTE	nnSMOTE	nbSMOTE	logregSMOTE	knnSMOTE	kNN
wisconsin	.9739 ± .0036	.9723 ± .0033	.9732 ± .0025	.9725 ± .0034	**.9759 ± .0021**	.9711 ± .0035	.9178 ± .0793	.9659 ± .0022	.9662 ± .0040	.9732 ± .0041	.9657 ± .0035
haberman	.6003 ± .0317	.6263 ± .0209	.6116 ± .0213	.6359 ± .0242	.4310 ± .0381	.5910 ± .0288	.4410 ± .1229	.5841 ± .0194	**.6403 ± .0244**	.6090 ± .0210	.5233 ± .0418
new-thyroid1	.9628 ± .0220	.9639 ± .0199	.9339 ± .0412	**.9853 ± .0055**	.8409 ± .0895	.9673 ± .0163	.7315 ± .2134	.9678 ± .0038	.9600 ± .0283	.9692 ± .0149	.9477 ± .0256
glass6	.9092 ± .0223	.9201 ± .0143	.8722 ± .0280	.8958 ± .0222	.8121 ± .0235	**.9271 ± .0115**	.6395 ± .1894	.8848 ± .0216	.8813 ± .0260	.8856 ± .0193	.8664 ± .0128
ecoli-0-4-6_vs_5	.8887 ± .0213	.8824 ± .0240	.7119 ± .1487	.8784 ± .0379	.8330 ± .0288	**.9091 ± .0277**	.7128 ± .1167	.8694 ± .0311	.8634 ± .0248	.8865 ± .0142	.8793 ± .0265
glass-0-1-6_vs_2	.7317 ± .0419	.5820 ± .0455	.6950 ± .0501	.7274 ± .0472	.6328 ± .0614	.6511 ± .0513	.5384 ± .0879	.5179 ± .0182	**.7627 ± .0455**	.6962 ± .0397	.5922 ± .0407
glass2	.7433 ± .0577	.5816 ± .0425	.7185 ± .0498	.7424 ± .0493	.5973 ± .0514	.6638 ± .0588	.5213 ± .0836	.5780 ± .0167	**.7755 ± .0308**	.7203 ± .0382	.6001 ± .0382
shuttle-c0-vs-c4	.9959 ± .0006	**1 ± 0**	.9850 ± .0110	.9915 ± .0230	.7424 ± .1712	**1 ± 0**	.9547 ± .0945	.9993 ± .0013	.9950 ± .0036	.9972 ± .0021	.9953 ± .0015
glass4	.9093 ± .0375	.8846 ± .0361	.8925 ± .0391	.8934 ± .0420	.8658 ± .0229	.9143 ± .0326	.8137 ± .1120	.7908 ± .0514	.8272 ± .0586	.9162 ± .0209	**.9191 ± .0375**
glass-0-1-6_vs_5	.8628 ± .0462	.8851 ± .0467	.8892 ± .0345	.8603 ± .0450	.7723 ± .0587	**.9618 ± .0273**	.9514 ± .0494	.7656 ± .0465	.9328 ± .0319	.9304 ± .0331	.8281 ± .0356
shuttle-c2-vs-c4	.9915 ± .0204	.9918 ± .0205	.7213 ± .1164	.8659 ± .1441	.5636 ± .1572	**.9930 ± .0192**	.9660 ± .0444	.9824 ± .0206	.9635 ± .0319	.9662 ± .0366	.9859 ± .0253
shuttle-6_vs_2-3	**1 ± 0**	**1 ± 0**	.6364 ± .1082	.6706 ± .1787	.4659 ± .1061	**1 ± 0**	.9681 ± .0441	.9048 ± .0412	.9333 ± .0229	**1 ± 0**	.9801 ± .0280
yeast-2_vs_8	**.7551 ± .0332**	.6453 ± .0559	.7381 ± .0224	.7274 ± .0173	.4998 ± .0360	.7362 ± .0262	.7371 ± .0411	.6704 ± .0439	.7479 ± .0352	.7266 ± .0289	.7158 ± .0200
yeast4	.8225 ± .0181	.7673 ± .0235	.8204 ± .0171	.8183 ± .0205	.7314 ± .0223	.8015 ± .0198	.7957 ± .0311	.6528 ± .0456	.8160 ± .0174	**.8303 ± .0164**	.5354 ± .0411
yeast-1-2-8-9_vs_7	.6782 ± .0378	.5495 ± .0444	.6568 ± .0344	.6901 ± .0295	.5405 ± .0306	.6473 ± .0449	.6751 ± .0378	.4238 ± .0486	**.7150 ± .0311**	.6665 ± .0299	.4640 ± .0461
yeast5	**.9642 ± .0163**	.9437 ± .0153	.9563 ± .0078	.9553 ± .0121	.9420 ± .0094	.9534 ± .0119	.9455 ± .0172	.9257 ± .0096	.9524 ± .0172	.9604 ± .0103	.7932 ± .0338
ecoli-0-1-3-7_vs_2-6	.8673 ± .0052	.8475 ± .0114	.7702 ± .0573	.8311 ± .0612	.6722 ± .0804	.8724 ± .0208	.7875 ± .0948	**.8913 ± .0254**	.8637 ± .0463	.8561 ± .0356	.8793 ± .0023
yeast6	.8599 ± .0160	.8259 ± .0187	**.8807 ± .0116**	.8719 ± .0178	.7725 ± .0294	.8475 ± .0189	.8362 ± .0226	.7746 ± .0279	.8576 ± .0128	.8508 ± .0142	.6995 ± .0281
winequality-white-3_vs_7	.6758 ± .0509	.6370 ± .0551	.5950 ± .1050	.6836 ± .0539	.6143 ± .0900	.6693 ± .0491	.5154 ± .1400	**.7570 ± .0342**	.6451 ± .0541	.6935 ± .0478	.5319 ± .0177
abalone-19_vs_10-11-12-13	.5784 ± .0415	.4894 ± .0350	.6080 ± .0325	.6361 ± .0384	.4826 ± .0328	.6082 ± .0481	.6112 ± .0985	.5354 ± .0164	**.6934 ± .0281**	.6338 ± .0374	*0 ± 0*
winequality-white-3-9_vs_5	.5369 ± .0454	.5299 ± .0435	.4918 ± .0674	.5229 ± .0498	.4499 ± .0451	.5876 ± .0481	.5272 ± .0890	**.7336 ± .0266**	.6432 ± .0480	.5289 ± .0544	*0 ± 0*
shuttle-2_vs_5	.9993 ± .0030	**1 ± 0**	.9809 ± .0100	.9884 ± .0039	.9143 ± .0594	**1 ± 0**	.9925 ± .0141	.9303 ± .0192	.9964 ± .0017	.9970 ± .0029	.9987 ± .0046
abalone-20_vs_8-9-10	.6669 ± .0588	.7395 ± .0395	.7618 ± .0409	.8198 ± .0290	.5847 ± .0508	.8156 ± .0336	.8474 ± .0334	.6912 ± .0119	**.8539 ± .0269**	.7764 ± .0386	.4776 ± .0289
poker-8-9_vs_5	**.7763 ± .0560**	.6074 ± .0492	.6160 ± .0879	.5447 ± .1050	.2369 ± .0847	.5217 ± .0501	.6039 ± .0490	.4681 ± .0455	.5002 ± .0386	.7515 ± .0409	.4482 ± .0089
abalone19	.6047 ± .0478	.4235 ± .0287	.6725 ± .0343	.7110 ± .0350	.5296 ± .0373	.6700 ± .0354	.6311 ± .0994	.6680 ± .0184	**.7280 ± .0265**	.6955 ± .0370	*0 ± 0*

The F1 scores obtained are presented in Table 11. When applying k-NN on some data sets, the classifier was only predicting

as the negative class, which made the F1 score as "NA", and we decided to report it as zero in the results table. Also in this case, as can be observed, the classifiers are performing differently on the data sets. Again, to check that the results are different in terms of ranking, we applied the Friedman test. As in the case of the GM scores, the degrees of freedom are 10 and for this case we obtained a value of 106.54 for the statistic, with a p-value of $2.20 \cdot 10^{-16}$. The results obtained indicate that also in this case there is a difference among the mean of the classifiers' average ranks at the chosen significance level of 0.05.

Table 11

F1 score for the selected KEEL data sets; the maximum F1 scores for each data set are displayed in bold and underlined

Data sets	VoronoiGA	XGBoost	svmUNDER	svmSMOTE	svmBOTH	rfSMOTE	nnSMOTE	nbSMOTE	logregSMOTE	knnSMOTE	kNN
wisconsin	.9597 ± .0043	.9600 ± .0036	**.9608 ± .0028**	.9592 ± .0039	.9607 ± .0030	.9585 ± .0039	.9256 ± .0353	.9477 ± .0027	.9546 ± .0046	.9602 ± .0047	.9558 ± .0041
haberman	.4438 ± .0361	.4723 ± .0237	.4714 ± .0236	.4857 ± .0273	.4327 ± .0155	.4388 ± .0342	.4552 ± .0286	.4404 ± .0227	**.4928 ± .0260**	.4538 ± .0237	.3739 ± .0467
new-thyroid1	.9319 ± .0284	.9400 ± .0288	.8544 ± .0491	.9396 ± .0144	.6640 ± .0845	**.9473 ± .0204**	.7804 ± .1474	.8667 ± .0130	.9261 ± .0367	.8941 ± .0336	.9364 ± .0290
glass6	.8341 ± .0409	.8479 ± .0311	.7273 ± .0465	.7979 ± .0315	.5125 ± .0340	**.8696 ± .0225**	.6236 ± .1256	.7928 ± .0270	.7648 ± .0376	.7807 ± .0359	.8179 ± .0251
ecoli-0-4-6_vs_5	.7958 ± .0528	.7415 ± .0392	.5256 ± .1171	.8077 ± .0388	.4733 ± .0366	.7817 ± .0415	.5113 ± .0894	.7243 ± .0554	.6532 ± .0438	.7077 ± .0347	**.8324 ± .0395**
glass-0-1-6_vs_2	.3477 ± .0406	.2792 ± .0604	.3082 ± .0427	.3702 ± .0382	.2524 ± .0343	.3447 ± .0480	.2138 ± .0369	.1809 ± .0101	**.4045 ± .0456**	.2994 ± .0322	.3802 ± .0503
glass2	.3417 ± .0486	.2749 ± .0439	.3023 ± .0394	.3711 ± .0437	.2133 ± .0256	.3491 ± .0561	.1979 ± .0316	.1879 ± .0076	**.4141 ± .0376**	.3002 ± .0313	.3935 ± .0481
shuttle-c0-vs-c4	.9951 ± .0023	**1 ± 0**	.8948 ± .0680	.9535 ± .0234	.6920 ± .1623	**1 ± 0**	.9533 ± .0847	.9921 ± .0059	.9939 ± .0046	.9971 ± .0022	.9953 ± .0015
glass4	.7566 ± .0740	.6386 ± .0496	.4924 ± .0665	.6563 ± .0825	.3637 ± .0390	.7222 ± .0545	.5716 ± .0977	.4408 ± .0719	.5419 ± .0803	.6151 ± .0497	**.8040 ± .0549**
glass-0-1-6_vs_5	.6497 ± .0634	.5304 ± .0639	.4457 ± .0469	.5821 ± .0855	.2211 ± .0209	.7367 ± .0762	**.7818 ± .0820**	.2384 ± .0469	.5540 ± .0764	.5327 ± .0765	.6836 ± .0600
shuttle-c2-vs-c4	.9877 ± .0252	.9907 ± .0234	.3404 ± .1385	.8215 ± .1377	.2634 ± .1132	**.9920 ± .0219**	.8987 ± .0900	.8593 ± .0805	.8932 ± .0822	.8810 ± .1155	.9840 ± .0288
shuttle-6_vs_2-3	**1 ± 0**	**1 ± 0**	.1922 ± .0509	.4019 ± .1400	.1137 ± .0105	**1 ± 0**	.9319 ± .0586	.8233 ± .0597	.8659 ± .0533	**1 ± 0**	.9773 ± .0319
yeast-2_vs_8	.2565 ± .0310	.4304 ± .0830	.5663 ± .0539	.6018 ± .0617	.0951 ± .0082	.5351 ± .0498	.4044 ± .0617	.3896 ± .1295	.4124 ± .0648	.3125 ± .0455	**.6269 ± .0369**
yeast4	.2921 ± .0172	**.3186 ± .0166**	.2688 ± .0164	.2874 ± .0150	.1405 ± .0089	.3175 ± .0160	.2630 ± .0178	.1336 ± .0272	.2686 ± .0146	.2650 ± .0146	.3183 ± .0404
yeast-1-2-8-9_vs_7	.1379 ± .0159	.1515 ± .0236	.1347 ± .0164	.1369 ± .0119	.0812 ± .0048	.1648 ± .0226	.1402 ± .0144	.0714 ± .0058	.1493 ± .0143	.1141 ± .0101	**.2423 ± .0475**
yeast5	.6142 ± .0264	.5747 ± .0185	.4556 ± .0174	.5157 ± .0190	.3761 ± .0240	**.6641 ± .0254**	.5806 ± .0282	.3570 ± .0230	.5626 ± .0192	.5003 ± .0198	.6561 ± .0440
ecoli-0-1-3-7_vs_2-6	.5461 ± .0661	.4201 ± .0769	.3205 ± .1094	.4955 ± .0937	.2424 ± .0624	.5173 ± .0819	.2885 ± .0645	.7380 ± .0844	.3085 ± .0764	.2958 ± .0708	**.7656 ± .0604**
yeast6	.2954 ± .0272	.3591 ± .0219	.3023 ± .0187	.2945 ± .0173	.1270 ± .0198	.3454 ± .0211	.2506 ± .0176	.1394 ± .0240	.2438 ± .0173	.2106 ± .0135	**.5085 ± .0341**
winequality-white-3_vs_7	.1055 ± .0144	.2871 ± .0529	.0991 ± .0236	.1051 ± .0140	.0780 ± .0094	.1693 ± .0260	.1181 ± .0353	.1884 ± .0236	.0946 ± .0138	.1059 ± .0138	**.3687 ± .0283**
abalone-19_vs_10-11-12-13	.0645 ± .0091	**.1087 ± .0177**	.0590 ± .0061	.0767 ± .0086	.0450 ± .0033	.0923 ± .0139	.0877 ± .0141	.0463 ± .0025	.0954 ± .0072	.0705 ± .0077	*0 ± 0*
winequality-white-3-9_vs_5	.0454 ± .0077	**.1778 ± .0289**	.0450 ± .0065	.0418 ± .0069	.0357 ± .0042	.0941 ± .0181	.0600 ± .0135	.1045 ± .0090	.0664 ± .0089	.0402 ± .0080	*0 ± 0*
shuttle-2_vs_5	.9859 ± .0154	**1 ± 0**	.7872 ± .0742	.9679 ± .0410	.4113 ± .1255	**1 ± 0**	.7964 ± .0411	.2776 ± .0748	.8259 ± .0359	.8789 ± .0238	.9983 ± .0054
abalone-20_vs_8-9-10	.1007 ± .0217	.2630 ± .0250	.0957 ± .0199	.1862 ± .0199	.0397 ± .0049	.1826 ± .0190	.2340 ± .0209	.0531 ± .0018	.2373 ± .0179	.1177 ± .0145	**.2801 ± .0313**
poker-8-9_vs_5	.1325 ± .0207	.2275 ± .0331	.0522 ± .0149	.0610 ± .0220	.0259 ± .0022	.0740 ± .0140	.0489 ± .0083	.0222 ± .0040	.0247 ± .0034	.0709 ± .0087	**.2534 ± .0184**
abalone19	.0291 ± .0041	**.0892 ± .0187**	.0299 ± .0033	.0404 ± .0042	.0201 ± .0016	.0464 ± .0043	.0430 ± .0064	.0271 ± .0014	.0462 ± .0032	.0358 ± .0039	*0 ± 0*

In Table 12 the average rankings, along with the adjusted p-values, are displayed, which also facilitates analysis of the results obtained in Table 10 and Table 11. As can be observed from Table 12, the VoronoiGA strategy has a ranking_{GM} of 3.78 and a ranking_{F1} of 4.90 when compared with the other 10 classifiers. This places the VoronoiGA strategy relative to the other classifiers in first place for the GM average Friedman rankings, and in fourth place for the F1 average Friedman rankings. Since we maximized the GM scores when performing the learning process we expected to get better results for the GM scores than the F1 scores. Even in this case, the position for the F1 average Friedman rankings is fourth, which indicates that the VoronoiGA strategy finds the positive class (as F1 score focuses on the TP) comparable to the other classifiers. In Table 11 the VoronoiGA strategy gets the rank of 1 for the shuttle-6_vs_2-3 data set.

If we analyze carefully the GM score results obtained in Table 10, we can observe that VoronoiGA has a rank of 1 for 4 data sets, namely, shuttle-6_vs_2-3, yeast-2_vs_8, yeast5 and poker-8-9_vs_5, where all these data sets have an imbalance ratio larger than 20. The classifier with the second-best ranking_{GM} is k-NN with SMOTE (knnSMOTE), and for this classifier there are only 2 data sets where the rank is 1 and from which 1 is equal to the GM score obtained by VoronoiGA (shuttle-6_vs_2-3).

In terms of APV_{GM}, the null hypothesis of means equality is rejected at a significance level of 0.05 when comparing VoronoiGA with the other four classifiers, namely, svmBOTH, nnSMOTE, nbSMOTE and kNN. There are two classifiers for APV_{F1}, namely, svmBOTH and nbSMOTE, that reject the means equality null hypothesis at the same level of significance 0.05.

A final observation drawn from Table 12 is that for some classifiers the positions of ranking_{GM} and ranking_{F1} are changed between the two measures. A good example of a sudden change is k-NN, where it is clear that it is much better at detecting the positives than the negatives, since F1 scores focus on TP. A similar behaviour can also be observed for XGBoost.

Table 12
Average Friedman rankings, positions and adjusted p-values (APVs) of the selected algorithms using Holm's procedure for the GM and F1 scores; the APVs at a significance level of 0.05 are highlighted

Algorithms	Position $ranking_{GM}$	$ranking_{GM}$	APV_{GM}	Position $ranking_{F1}$	$ranking_{F1}$	APV_{F1}
VoronoiGA	1	3.78	-	4	4.90	-
XGBoost	6	5.98	.590	2	3.34	1.000
svmUNDER	7	6.30	.238	9	7.40	.231
svmSMOTE	5	4.84	1.000	5	5.32	1.000
svmBOTH	11	9.40	**$1.147 \cdot 10^{-7}$**	11	10.08	**$1.744 \cdot 10^{-6}$**
rfSMOTE	3	4.14	1.000	1	3.26	1.000
nnSMOTE	9	7.60	**.002**	8	6.96	.759
nbSMOTE	8	7.40	**.005**	10	8.44	**$6.596 \cdot 10^{-3}$**
logregSMOTE	4	4.60	1.000	6	5.80	1.000
knnSMOTE	2	3.88	1.000	7	6.66	1.000
kNN	10	8.08	**$2.283 \cdot 10^{-4}$**	3	3.84	1.000

To facilitate a better overview of the rankings obtained we decided to plot the GM score versus the F1 score average rankings in Fig. 9. As can also be seen in the figure, the VoronoiGA has the highest rank on the average ranking GM score axis, while on the average ranking F1 score there are three other classifiers that perform better. These results indicate, in our opinion, that VoronoiGA can indeed also be used for problems without an attached cost.

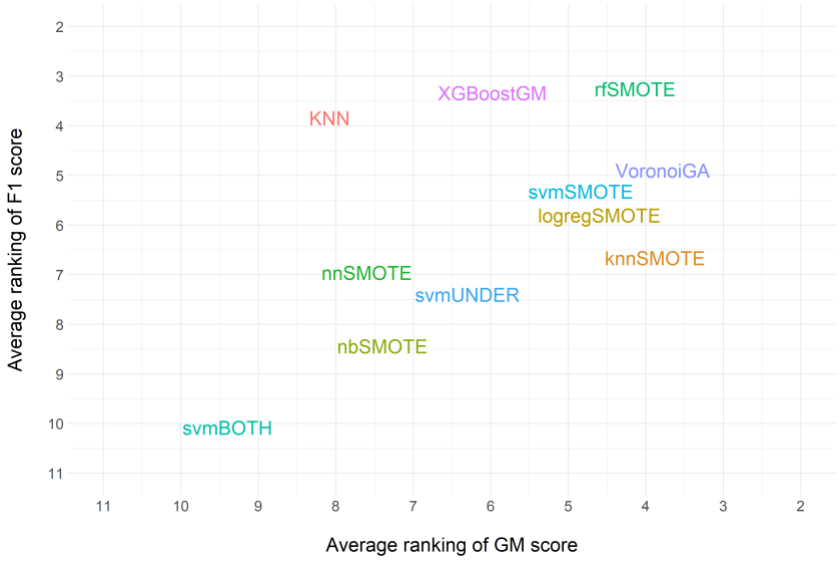


Fig. 9 Average ranking of GM score versus average ranking of F1 score

In Table 13 the running times for all the classifiers using the HPC cluster are displayed. We underlined and presented in bold the fastest-running classifiers. As in the industrial problem, since the naïve Bayes and logistic regression algorithms do not have parameters to tune, it is obvious that they are performing the fastest of all the classifiers.

Some data sets have a higher running time, since it was more difficult to find results where the classifier was not only predicting as the negative class. This was the case where the number of positive samples was very small. For k-NN, since the classifier was always classifying the observations as negative, we decided to report the running time as "Inf".

Similar to the industrial data set results, the Voronoi strategy runs faster than XGBoost and for some data sets it is also faster than the artificial neural networks model. There are also two cases of data sets where the support vector machine is running slower than VoronoiGA, namely, ecoli-0-1-3-7_vs_2-6 (svmSMOTE) and yeast6 (svmBOTH).

Table 13
Running time in seconds of the KEEL real-world data sets; the fastest-running classifiers for each data set are presented in bold and underlined

Data sets	VoronoiGA	XGBoost	svmUNDER	svmSMOTE	svmBOTH	rfSMOTE	nnSMOTE	nbSMOTE	logregSMOTE	knnSMOTE	kNN
wisconsin	303 ± 10	1946 ± 64	212 ± 20	273 ± 10	220 ± 20	179 ± 22	254 ± 14	.97 ± .03	**.89 ± .02**	105 ± 5	97 ± 4
haberman	475 ± 10	2133 ± 55	242 ± 30	223 ± 12	171 ± 15	128 ± 23	16870 ± 2661	.48 ± .02	**.47 ± .02**	85 ± 5	98 ± 6
new-thyroid1	680 ± 16	2166 ± 68	226 ± 21	240 ± 37	208 ± 10	93 ± 8	117 ± 5	**.42 ± .01**	.44 ± .01	92 ± 5	100 ± 9
glass6	421 ± 10	2121 ± 71	223 ± 16	229 ± 13	218 ± 18	111 ± 22	330 ± 87	**.45 ± .01**	.46 ± .02	86 ± 4	91 ± 4
ecoli-0-4-6_vs_5	493 ± 13	2185 ± 77	234 ± 23	236 ± 15	334 ± 34	111 ± 23	138 ± 14	.43 ± .02	**.41 ± .01**	80 ± 5	110 ± 3
glass-0-1-6_vs_2	514 ± 12	3214 ± 945	210 ± 11	222 ± 49	209 ± 21	96 ± 11	2182 ± 287	.46 ± .16	**.42 ± .03**	93 ± 11	140 ± 38
glass2	527 ± 13	2855 ± 707	238 ± 55	218 ± 33	193 ± 21	124 ± 40	2740 ± 722	**.43 ± .02**	.46 ± .07	89 ± 4	127 ± 23
shuttle-c0-vs-c4	619 ± 15	10221 ± 2810	228 ± 26	232 ± 19	166 ± 9	2 ± 0	128 ± 9	1.08 ± .03	**.73 ± .05**	100 ± 8	100 ± 8
glass4	875 ± 31	2187 ± 55	227 ± 7	250 ± 148	235 ± 21	88 ± 9	143 ± 26	**.43 ± .02**	.44 ± .04	90 ± 4	86 ± 4
glass-0-1-6_vs_5	539 ± 44	2337 ± 140	231 ± 19	196 ± 14	227 ± 25	121 ± 75	125 ± 12	.48 ± .23	**.44 ± .05**	88 ± 3	102 ± 15
shuttle-c2-vs-c4	680 ± 79	2342 ± 715	216 ± 11	517 ± 175	203 ± 13	115 ± 16	129 ± 16	**.40 ± .02**	**.40 ± .02**	99 ± 7	101 ± 14
shuttle-6_vs_2-3	844 ± 159	2062 ± 98	228 ± 17	174 ± 20	206 ± 22	182 ± 59	117 ± 7	.43 ± .04	**.42 ± .03**	155 ± 46	104 ± 7
yeast-2_vs_8	311 ± 15	2149 ± 211	190 ± 11	195 ± 18	212 ± 7	139 ± 112	334 ± 137	.46 ± .06	**.41 ± .03**	90 ± 5	107 ± 4
yeast4	255 ± 18	2214 ± 76	187 ± 14	209 ± 21	222 ± 8	119 ± 19	820 ± 171	.77 ± .03	**.46 ± .02**	91 ± 6	133 ± 89
yeast-1-2-8-9_vs_7	225 ± 13	2268 ± 170	196 ± 11	181 ± 30	206 ± 19	114 ± 23	641 ± 66	.67 ± .04	**.43 ± .04**	83 ± 6	173 ± 68
yeast5	699 ± 14	2018 ± 76	173 ± 10	210 ± 24	228 ± 19	115 ± 20	202 ± 13	.84 ± .04	**.44 ± .02**	91 ± 5	107 ± 3
ecoli-0-1-3-7_vs_2-6	972 ± 45	2216 ± 420	216 ± 23	1083 ± 516	270 ± 60	96 ± 17	286 ± 209	1.04 ± .90	**.39 ± .03**	107 ± 23	109 ± 8
yeast6	222 ± 15	2109 ± 78	194 ± 13	197 ± 26	232 ± 13	104 ± 12	329 ± 41	.79 ± .04	**.47 ± .03**	90 ± 6	111 ± 9
winequality-white-3_vs_7	482 ± 50	2236 ± 437	221 ± 19	191 ± 16	186 ± 32	126 ± 63	260 ± 38	.72 ± .03	**.43 ± .03**	82 ± 5	167 ± 133
abalone-19_vs_10-11-12-13	557 ± 38	2701 ± 1049	210 ± 7	177 ± 28	212 ± 14	122 ± 25	1907 ± 227	.88 ± .05	**.49 ± .06**	90 ± 6	*Inf*
winequality-white-3-9_vs_5	299 ± 29	2617 ± 551	224 ± 25	200 ± 34	196 ± 17	110 ± 16	791 ± 223	.87 ± .04	**.46 ± .03**	88 ± 10	*Inf*
shuttle-2_vs_5	969 ± 22	2096 ± 402	206 ± 21	227 ± 11	165 ± 15	548 ± 121	141 ± 9	1.35 ± .02	**.69 ± .08**	96 ± 5	140 ± 22
abalone-20_vs_8-9-10	892 ± 22	2209 ± 117	207 ± 10	190 ± 22	216 ± 19	104 ± 20	207 ± 20	.96 ± .04	**.47 ± .05**	92 ± 4	137 ± 34
poker-8-9_vs_5	494 ± 12	2359 ± 502	218 ± 21	273 ± 85	231 ± 22	129 ± 24	672 ± 232	1.07 ± .12	**.44 ± .04**	102 ± 6	297 ± 113
abalone19	883 ± 17	4269 ± 1675	209 ± 10	176 ± 34	212 ± 15	115 ± 23	1197 ± 249	1.60 ± .02	**.43 ± .02**	90 ± 7	*Inf*

8. Conclusion and final remarks

In this article the main focus has been on solving a real-life industrial problem that deals with the production of frequency inverter drives, which is a multi-stage manufacturing process. The product selected for the analysis contains two sub-assemblies (control card and power card), which are combined and tested by a unit test, which is the final quality assurance test in the manufacturing process. The problem is a predictive maintenance one, where we would like to predict the final quality of the product before it is sent to the client given the historical quality data.

For this, we first performed a feature engineering step where we added six newly engineered features, which are based on process knowledge and information hidden in terms of "NA". Next, we used XGBoost to lower the high-dimensional space to a two-dimensional one, where each axis represents a projection in terms of process sub-assemblies. The projection was obtained by carrying out a linear combination between the feature data and the gain percentages for each feature computed using XGBoost. For the feature selection process, it is important to mention that the number of unique values in a row is the feature with the highest gain percentage, which is in fact a newly engineered feature.

The data coming from the feature engineering process was further used in order to see how well the good and faulty products can be predicted. For this particular data set, the faulty product data represents 1%, while the good product data represents 99%, which means that the problem is a highly imbalanced one. From the industrial perspective, the current problem is formulated as a cost problem, where the cost of a faulty product sent to the client is approximately 20 times higher than the cost of a good product and where the industrial partner is also interested in the "quality" (number of faulty products sent to the client).

In order to incorporate the cost, "quality" and the imbalance into the prediction model, we modified an imbalanced learning strategy proposed by Khan, Schiøler, Zaki, and Kulahci (2018), which takes into consideration the imbalance, small disjuncts and overlap (noise) and is flexible enough to include the cost. The strategy is based on Voronoi diagrams combined with a genetic algorithm. For the actual implementation of the method, we created a fitness/objective function that is tuned by a chosen parameter, α, which spans between cost (in terms of checked products) and "quality" (1 – Sensitivity).

The modified strategy (VoronoiGA) was then compared with the other 10 base classifiers, most of which are run with a resampling strategy on the data. The obtained results indicate that VoronoiGA is flexible and performs much better than the other classifiers.

Since the industrial problem is formulated as a cost problem, we also wanted to show that VoronoiGA is flexible enough to perform in cases where no cost is specified. Therefore, we used 25 real-world data sets with different imbalance ratios from the KEEL repository, and for comparison we used the same classifiers as in the case of the industrial problem. As a performance measure, we focused on the Geometric Mean score, as it is a good imbalance measure and it also focuses on

the negative and positive classes in the same degree. We also decided to report the F1 score, which focuses mostly on the positive class, and it is also a common measure in the imbalance research literature. When compared with the other classifiers, the results of the analysis show that VoronoiGA ranks first for GM score average rankings and fourth for F1 score average rankings. This is a good indicator that the proposed modified strategy is also flexible enough to be used in problems where the cost is not specified.

To conclude, given the results obtained, we believe that the proposed strategy can be used successfully for imbalanced classification problems that are affected by small disjuncts, overlapping and noise.

9. Acknowledgements

For making this work possible, the authors of the article would like to thank MADE SPIR – Strategic Platform for Innovation and Research, Denmark.

10. References

Lasi, H., Fettke, P., Kemper, H. G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. Business & Information Systems Engineering, 6(4), 239-242. https://doi.org/10.1007/s12599-014-0334-4.

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer networks, 54(15), 2787-2805. https://doi.org/10.1007/s10796-014-9492-7.

Xiong, Y. L., & Yin, Z. P. (2006). Digital manufacturing—the development direction of the manufacturing technology in the 21 st century. Frontiers of Mechanical Engineering in China, 1(2), 125-130. https://doi.org/10.1007/s11465-006-0021-3.

Lee, J., Lapira, E., Bagheri, B., & Kao, H. A. (2013). Recent advances and trends in predictive manufacturing systems in big data environment. Manufacturing letters, 1(1), 38-41. https://doi.org/10.1016/j.mfglet.2013.09.005.

Lee, J., Lapira, E., Yang, S., & Kao, A. (2013). Predictive manufacturing system-Trends of next-generation production systems. IFAC Proceedings Volumes, 46(7), 150-156. https://doi.org/10.3182/20130522-3-BR-4036.00107.

Krumeich, J., Jacobi, S., Werth, D., & Loos, P. (2014, June). Big data analytics for predictive manufacturing control - A case study from process industry. In 2014 IEEE International Congress on Big Data (pp. 530-537). IEEE. https://doi.org/10.1109/BigData.Congress.2014.83.

Taguchi, G. (1986). Introduction to quality engineering: designing quality into products and processes.

Khan, A. R., Schiøler, H., Zaki, M., & Kulahci, M. (2018, June). Rare-Events Classification: An Approach Based on Genetic Algorithm and Voronoi Tessellation. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 256-266). Springer, Cham. https://doi.org/10.1007/978-3-030-04503-6_26.

das Chagas Moura, M., Zio, E., Lins, I. D., & Droguett, E. (2011). Failure and reliability prediction by support vector machines regression of time series data. Reliability Engineering & System Safety, 96(11), 1527-1534. https://doi.org/10.1016/j.ress.2011.06.006.

Kusiak, A. (2001). Rough set theory: a data mining tool for semiconductor manufacturing. IEEE transactions on electronics packaging manufacturing, 24(1), 44-50. https://doi.org/10.1109/6104.924792.

Iannuzzelli, R. (1991, May). Predicting plated-through-hole reliability in high temperature manufacturing processes. In 1991 Proceedings 41st Electronic Components & Technology Conference (pp. 410-421). IEEE. https://doi.org/10.1109/ECTC.1991.163908.

Denson, W. (1998). The history of reliability prediction. IEEE Transactions on Reliability, 47(3), 321–328. https://doi.org/10.1109/24.740547.

O'Connor, P., & Kleyner, A. (2011). Practical reliability engineering. (5th ed.). John Wiley & Sons. https://doi.org/10.1002/9781119961260.

Blischke, W. R., & Murthy, D. P. (2011). Reliability: modeling, prediction, and optimization (Vol. 767). John Wiley & Sons. https://doi.org/10.1002/9781118150481.

Harding, J. A., Shahbaz, M., & Kusiak, A. (2006). Data mining in manufacturing: a review. Journal of Manufacturing Science and Engineering, 128(4), 969-976. https://doi.org/10.1115/1.2194554.

Lee, J. H., & Park, S. C. (2001). Data mining for high quality and quick response manufacturing. In D. Braha, (Ed.), Data Mining for Design and Manufacturing (pp. 179-205). Springer, Boston, MA. https://doi.org/10.1007/978-1-4757-4911-3.

Sebzalli, Y. M., & Wang, X. Z. (2001). Knowledge discovery from process operational data using PCA and fuzzy clustering. Engineering Applications of Artificial Intelligence, 14(5), 607-616. https://doi.org/10.1016/S0952-1976(01)00032-X.

Fountain, T., Dietterich, T., & Sudyka, B. (2003, January). Data mining for manufacturing control: an application in optimizing IC tests. In G. Lakemeyer, B. Nebel, (Eds.), Exploring Artificial Intelligence in the New Millennium (pp. 381-400). Morgan Kaufmann Publishers Inc.

Cateni, S., Colla, V., & Vannucci, M. (2014). A method for resampling imbalanced datasets in binary classification tasks for real-world problems. Neurocomputing, 135, 32-41. http://doi.org/10.1016/j.neucom.2013.05.059.

Kusiak, A., & Kurasek, C. (2001). Data mining of printed-circuit board defects. IEEE transactions on robotics and automation, 17(2), 191-196. https://doi.org/10.1109/70.928564.

Chen, W. C., Lee, A. H., Deng, W. J., & Liu, K. Y. (2007). The implementation of neural network for semiconductor PECVD process. Expert Systems with Applications, 32(4), 1148-1153. https://doi.org/10.1016/j.eswa.2006.02.013.

Kim, A., Oh, K., Jung, J. Y., & Kim, B. (2018). Imbalanced classification of manufacturing quality conditions using cost-sensitive decision tree ensembles. International Journal of Computer Integrated Manufacturing, 31(8), 701-717. https://doi.org/10.1080/0951192X.2017.1407447.

Khan, A. R., Schiøler, H., Knudsen, T., & Kulahci, M. (2015, September). Statistical data mining for efficient quality control in manufacturing. In 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA) (pp. 1-4). IEEE. https://doi.org/10.1109/ETFA.2015.7301625.

Khan, A. R., Schiøler, H., Kulahci, M., & Knudsen, T. (2017, September). Big data analytics for industrial process control. In 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1-8). IEEE. https://doi.org/10.1109/ETFA.2017.8247658.

Köksal, G., Batmaz, İ., & Testik, M. C. (2011). A review of data mining applications for quality improvement in manufacturing industry. Expert Systems with Applications, 38(10), 13448-13467. https://doi.org/10.1016/j.eswa.2011.04.063.

He, H. & Ma, Y. (2013). Imbalanced learning. Wiley. https://doi.org/10.1002/9781118646106.

Sun, Y., Wong, A. K., & Kamel, M. S. (2009). Classification of imbalanced data: A review. International Journal of Pattern Recognition and Artificial Intelligence, 23(04), 687-719. https://doi.org/10.1142/S0218001409007326.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. Expert Systems with Applications, 73, 220-239. https://doi.org/10.1016/j.eswa.2016.12.035.

Liu, X. Y., & Zhou, Z. H. (2006). The influence of class imbalance on cost-sensitive learning: An empirical study. In Sixth International Conference on Data Mining (ICDM'06), IEEE, 970-974. https://doi.org/10.1109/icdm.2006.158

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357. https://doi.org/10.1613/jair.953.

Domingos, P. (1999, August). Metacost: A general method for making classifiers cost-sensitive. In KDD '99 Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (Vol. 99, pp. 155-164). https://doi.org/10.1145/312129.312220.

Elkan, C. (2001, August). The foundations of cost-sensitive learning. In Proceeding IJCAI'01 Proceedings of the 17th international joint conference on Artificial Intelligence (Vol. 17, No. 1, pp. 973-978). Lawrence Erlbaum Associates Ltd.

Napierała, K., & Stefanowski, J. (2012, March). Identification of different types of minority class examples in imbalanced data. In E. Corchado, V. Snášel, A. Abraham, M. Wozniak, M. Graña, S.-B. Cho (Eds.), International Conference on Hybrid Artificial Intelligence Systems (pp. 139-150). Springer, Berlin, Heidelberg.

Japkowicz, N. (2003, August). Class imbalances: are we focusing on the right issue. In Workshop on Learning from Imbalanced Data Sets II (Vol. 1723, p. 63).

Jo, T., & Japkowicz, N. (2004). Class imbalances versus small disjuncts. ACM Sigkdd Explorations Newsletter, 6(1), 40-49. https://doi.org/10.1145/1007730.1007737.

Lee, H. K., & Kim, S. B. (2018). An overlap-sensitive margin classifier for imbalanced and overlapping data. Expert Systems with Applications, 98, 72-83. https://doi.org/10.1016/j.eswa.2018.01.008.

García, V., Mollineda, R. A., & Sánchez, J. S. (2008). On the k-NN performance in a challenging scenario of imbalance and overlapping. Pattern Analysis and Applications, 11(3-4), 269-280. https://doi.org/10.1007/s10044-007-0087-5.

Van Hulse, J., & Khoshgoftaar, T. (2009). Knowledge discovery from imbalanced and noisy data. Data & Knowledge Engineering, 68(12), 1513-1542. https://doi.org/10.1016/j.datak.2009.08.005.

Napierała, K., Stefanowski, J., & Wilk, S. (2010, June). Learning from imbalanced data in presence of noisy and borderline examples. In M. Szczuka, M. Kryszkiewicz, S. Ramanna, R. Jensen, Q. Hu (Eds.), International Conference on Rough Sets and Current Trends in Computing (pp. 158-167). Springer, Berlin, Heidelberg.

Lee, D. T., & Schachter, B. J. (1980). Two algorithms for constructing a Delaunay triangulation. International Journal of Computer & Information Sciences, 9(3), 219-242. https://doi.org/10.1007/BF00977785.

Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464-1480. https://doi.org/10.1109/5.58325.

Nova, D., & Estévez, P. A. (2014). A review of learning vector quantization classifiers. Neural Computing and Applications, 25(3-4), 511-524. http://doi.org/10.1007/s00521-013-1535-3.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. Machine learning, 3(2), 95-99. https://doi.org/10.1023/A:1022602019183.

Clopper, C. J., & Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika, 26(4), 404-413.

Agresti, A., & Coull, B. A. (1998). Approximate is better than “exact” for interval estimation of binomial proportions. The American Statistician, 52(2), 119-126.

Grbovic, M., & Vucetic, S. (2009, June). Learning vector quantization with adaptive prototype addition and removal. In 2009 International Joint Conference on Neural Networks (pp. 994-1001). IEEE, https://doi.org/10.1109/IJCNN.2009.5178710.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal

Statistical Society. Series C (Applied Statistics), 28(1), 100-108. https://doi.org/10.2307/2346830.

Mitchell, T.M. (1997). Machine Learning. (1st ed.). McGraw-Hill

Pestov, V. (2013). Is the k-NN classifier in high dimensions affected by the curse of dimensionality?. Computers & Mathematics with Applications, 65(10), 1427-1437. https://doi.org/10.1016/j.camwa.2012.09.011.

Indyk, P., & Motwani, R. (1998, May). Approximate nearest neighbors: towards removing the curse of dimensionality. In Proceedings of the thirtieth annual ACM symposium on Theory of computing (pp. 604-613). ACM.

Hand, D. J. (2009). Measuring classifier performance: a coherent alternative to the area under the ROC curve. Machine learning, 77(1), 103-123. https://doi.org/10.1007/s10994-009-5119-5.

Ferri, C., Hernández-Orallo, J., & Flach, P. A. (2011). A coherent interpretation of AUC as a measure of aggregated classification performance. In Proceedings of the 28th International Conference on Machine Learning (ICML-11) (pp. 657-664).

García, S., Triguero, I., Carmona, C. J., & Herrera, F. (2012). Evolutionary-based selection of generalized instances for imbalanced classification. Knowledge-Based Systems, 25(1), 3-12. https://doi.org/10.1016/j.knosys.2011.01.012

DTU HPC Cluster (LSF 10). (2019). https://www.hpc.dtu.dk/?page_id=2520 Accessed 13 May 2019

Friedman, J. H. (2002). Stochastic gradient boosting. Computational Statistics and Data Analysis, 38(4), 367-378. https://doi.org/10.1016/S0167-9473(01)00065-2.

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.

XGBoost Parameters. (2019). https://xgboost.readthedocs.io/en/latest/parameter.html Accessed 13 May 2019

Cawley, G. C., & Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. Journal of Machine Learning Research, 11(Jul), 2079-2107.

Krstajic, D., Buturovic, L. J., Leahy, D. E., & Thomas, S. (2014). Cross-validation pitfalls when selecting and assessing regression and classification models. Journal of Cheminformatics, 6(1), 10. https://doi.org//10.1186/1758-2946-6-10.

Wainer, J., & Cawley, G. (2018). Nested cross-validation when selecting classifiers is overzealous for most practical applications. arXiv preprint arXiv:1809.09446. https://arxiv.org/abs/1809.09446.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In Advances in Neural Information Processing Systems (pp. 2951-2959).

Frazier, P. I. (2018). A tutorial on bayesian optimization. arXiv preprint arXiv:1807.02811. https://arxiv.org/abs/1807.02811.

Yu, H., Mu, C., Sun, C., Yang, W., Yang, X., & Zuo, X. (2015). Support vector machine-based optimized decision threshold adjustment strategy for classifying imbalanced data. Knowledge-Based Systems, 76, 67-78. http://doi.org/10.1016/j.knosys.2014.12.007.

Vapnik, V.N. (1998) Statistical learning theory. (1st ed.). Wiley - Interscience, New York.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics. https://doi.org/10.1007/978-0-387-84858-7.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Computing, 13(10), 959.

https://doi.org/10.1007/s00500-008-0392-y.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences, 180(10), 2044-2064. https://doi.org/10.1016/j.ins.2009.12.010.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2011). KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing, 17, 255–287.